Document Title	Specification of Ethernet Driver
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	430

Document Status	Final
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	4.4.0

Document Change History

Date	Release	Changed by	Change Description
2018-10-31	4.4.0	AUTOSAR Release Management	• Support of host controllers with multiple cores • Asynchronous frame transmission • Timestamp improvements • Multicast MAC address handling in Switches
2017-12-08	4.3.1	AUTOSAR Release Management	• Minor adaptions and corrections
2016-11-30	4.3.0	AUTOSAR Release Management	• Quality of Service (QoS) support • Ethernet statistics counter access
2015-07-31	4.2.2	AUTOSAR Release Management	• Eth_ControllerInit functionality merged into Eth_Init API • Development Error Tracer renamed to Default Error Tracer • IRQ handler API removed
2014-10-31	4.2.1	AUTOSAR Release Management	• Change from Synchronous to Asynchronous API • gPTP Timestamp Support • Enhanced Production Errors • Changed Access to Statistic Frame Handling Registers
2014-03-31	4.1.3	AUTOSAR Release Management	• Introduction of periodic call to Eth_SetControllerMode • Support of VLANs (Virtual Local Area Networks) • Editorial changes

Document Change History			
Date	**Release**	**Changed by**	**Change Description**
2013-10-31	4.1.2	AUTOSAR Release Management	• Introduction of Eth_GeneralTypes.h • Support of API deviation for asynchronous implementation • Changes in API of EthIf_ProvideTxBuffer and EthIf_SetPhysAddr • Editorial changes • Removed chapter(s) on change documentation
2013-03-15	4.1.1	AUTOSAR Administration	• Configurable MAC address based filtering • Detection of lost Ethernet frames • Buffer handling enhancement
2011-12-22	4.0.3	AUTOSAR Administration	• Description of buffer behaviour in Eth_SetControllerMode extended.
2010-09-30	3.1.5	AUTOSAR Administration	• Enhanced development error detection for active controller before controller access • Further post-build configurable parameters • Improved description of 'XxxCtrlIdx' semantics • 'Instance ID' removed from Version Info (concerns Eth_GetVersionInfo API) • Additional development error in Eth_GetVersionInfo API
2010-02-02	3.1.4	AUTOSAR Administration	• Initial Release

Disclaimer

Table of Contents

Known Limitations
Currently, chapter 5 Dependencies to other modules does not describe the versions of dependent modules. Thus, a version check will extend the chapter.

1 Introduction and functional overview

This specification specifies the functionality, API and the configuration of the AUTOSAR Basic Software module Ethernet Driver.

In the AUTOSAR Layered Software Architecture, the Ethernet Driver belongs to the *Microcontroller Abstraction Layer*, or more precisely, to the *Communication Drivers*.

This indicates the main task of the Ethernet Driver:
Provide to the upper layer (Ethernet Interface) a hardware independent interface comprising multiple equal controllers. This interface shall be uniform for all controllers. Thus, the upper layer (Ethernet Interface) may access the underlying bus system in a uniform manner. The interface provides functionality for initialization, configuration and data transmission. The configuration of the Ethernet Driver however is bus specific, since it takes into account the specific features of the communication controller.

A single Ethernet Driver module supports only one type of controller hardware, but several controllers of the same type. Additionally, the Ethernet Driver has to be able to be interoperable with the Switch Driver, if it is in a managed mode. In this case, a special treatment of the Ethernet frame might be necessary to fit a specific interpretation by a Switch device afterwards. The Ethernet Driver's prefix requires a unique namespace. The Ethernet Interface can access different controller types using different Ethernet Drivers using this prefix. The decision which driver to use to access a particular controller is a configuration parameter of the Ethernet Interface.

Figure 1.1 depicts the lower part of the Ethernet stack. One Ethernet Interface accesses several controllers using one or several Ethernet Drivers.

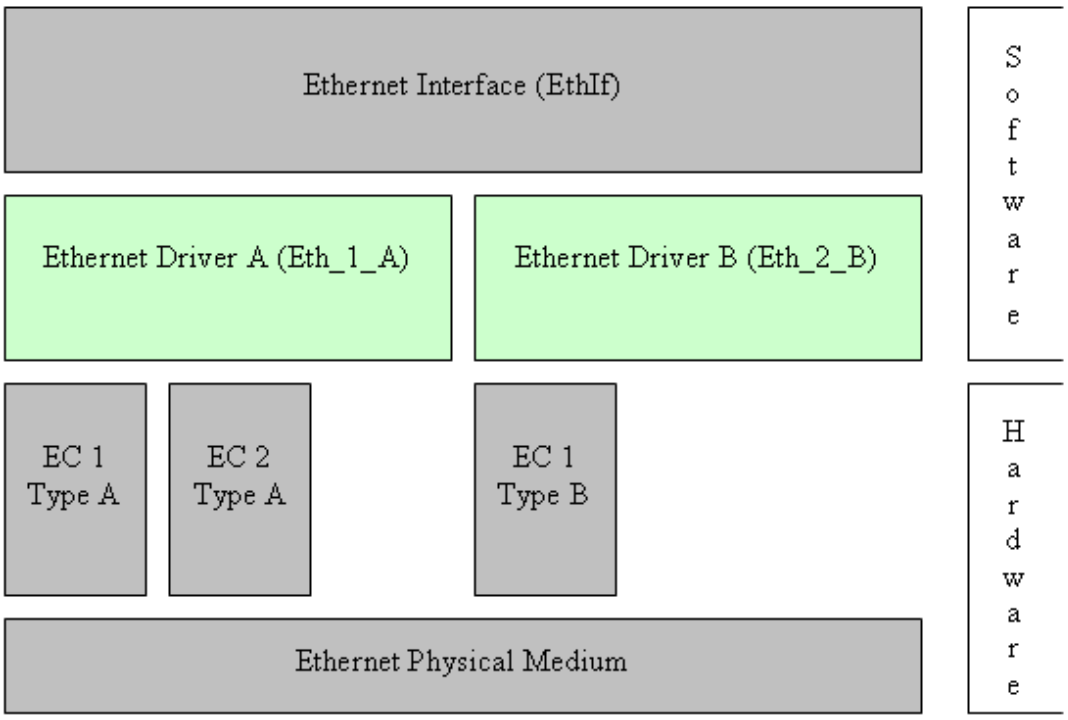

Figure 1.1: Ethernet stack module overview

Note: The Ethernet Driver is specified in a way that allows for object code delivery of the code module, following the "one-fits-all" principle, i.e. the entire configuration of the Ethernet Interface can be carried out without modifying any source code. Thus, the configuration of the Ethernet Driver can be carried out largely without detailed knowledge of the Ethernet Driver software.

2 Acronyms and abbreviations

Abbreviation / Acronym:	*Description:*
EC	Ethernet controller
Eth	Ethernet Driver (AUTOSAR BSW module)
EthIf	Ethernet Interface (AUTOSAR BSW module)
EthTrcv	Ethernet Transceiver Driver (AUTOSAR BSW module)
ISR	Interrupt Service Routine
MCG	Module Configuration Generator
MII	Media Independent Interface (standardized Interface provided by Ethernet controllers to access Ethernet transceivers)
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

3 Related documentation

3.1 Input documents

[1] List of Basic Software Modules
AUTOSAR_TR_BSWModuleList.pdf

[2] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

[3] AUTOSAR General Requirements on Basic Software Modules
AUTOSAR_SRS_BSWGeneral.pdf

[4] Specification of Communication
AUTOSAR_SWS_COM.pdf

[5] Requirements on Ethernet Support in AUTOSAR
AUTOSAR_SRS_Ethernet.pdf

[6] Specification of Ethernet Interface
AUTOSAR_SWS_EthernetInterface.pdf

[7] Specification of Ethernet State Manager
AUTOSAR_SWS_EthernetStateManager.pdf

[8] Specification of Ethernet Transceiver Driver
AUTOSAR_SWS_EthernetTransceiver.pdf

[9] Specification of Socket Adapter
AUTOSAR_SWS_SocketAdapter.pdf

[10] Specification of UDP Network Management
AUTOSAR_SWS_UDPNetworkManagement.pdf

[11] Specification of PDU Router
AUTOSAR_SWS_PDURouter.pdf

[12] BSW Scheduler Specification
AUTOSAR_SWS_Scheduler.pdf

[13] Specification of ECU Configuration
AUTOSAR_TPS_ECUConfiguration.pdf

[14] Specification of Memory Mapping
AUTOSAR_SWS_MemoryMapping.pdf

[15] Specification of Standard Types
AUTOSAR_SWS_StandardTypes.pdf

[16] Specification of Default Error Tracer
AUTOSAR_SWS_DefaultErrorTracer.pdf

[17] Specification of Diagnostics Event Manager
AUTOSAR_SWS_DiagnosticEventManager

[18] Specification of ECU State Manager
AUTOSAR_SWS_ECUStateManager.pdf

[19] General Specification of Basic Software Modules
AUTOSAR_SWS_BSWGeneral.pdf

3.2 Related standards and norms

[20] IEEE 802.3-2006

[21] IEC 7498-1 The Basic Model, IEC Norm, 1994

[22] IETF RFC 2819

[23] IEEE Standard 802.1AS™- 30 of March 2011
http://standards.ieee.org/getieee802/download/802.1AS-2011.pdf

3.3 Related specification

AUTOSAR provides a General Specification on Basic Software modules [19] (SWS BSW General), which is also valid for Ethernet Driver.

Thus, the specification SWS BSW General shall be considered as additional and required specification for Ethernet Driver.

4 Constraints and assumptions

4.1 Limitations

The Ethernet Driver module is only able to handle a single thread of execution. The execution must not be pre-empted by itself.

It is not possible to transmit data which exceeds the available buffer size of the used controller. Longer data has to be transmitted using the Internet Protocol (IP) or Transmission Control Protocol (TCP).

Depending on the Ethernet hardware, it may become necessary that implementations deviate from API specifications in respect to the asynchronous/synchronous behaviour.

4.2 Applicability to car domains

The Ethernet BSW stack is intended to be used wherever high data rates are required but no hard real-time is required. Of course, it can also be used for less-demanding use cases, i.e. for low data rates.

5 Dependencies to other modules

This chapter lists the modules interacting with the Ethernet Driver module.

Modules that use Ethernet Driver module:

- Ethernet Interface (EthIf)
- Ethernet Transceiver Driver (EthTrcv)

Modules used by the Ethernet Driver module:

- BSW Scheduler mechanisms for data consistency and main function handling.

Dependencies to other Modules:

- On certain systems the controller might share resources with other components (e.g. the MCU, Port), and may depend on their configuration. If those resources are within scope of the other modules (e.g. PLL configuration, memory mapping, etc.) the Ethernet Driver module does not take care of configuring those components but requires their preceding initialization.

6 Requirements traceability

Requirement	Description	Satisfied by
SRS_BSW_00101	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	SWS_Eth_00248, SWS_Eth_00252
SRS_BSW_00323	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	SWS_Eth_00249, SWS_Eth_00250, SWS_Eth_00253, SWS_Eth_00254
SRS_BSW_00369	All AUTOSAR Basic Software Modules shall not return specific development error codes via the API	SWS_Eth_00249, SWS_Eth_00250, SWS_Eth_00253, SWS_Eth_00254
SRS_BSW_00416	The sequence of modules to be initialized shall be configurable	SWS_Eth_00248, SWS_Eth_00252
SRS_Eth_00053	SWS shall specify configuration	SWS_Eth_00251, SWS_Eth_00255
SRS_ETH_00086	-	SWS_Eth_91001
SRS_Eth_00127	The Ethernet Driver shall provide statistic counter values	SWS_Eth_00026, SWS_Eth_00226, SWS_Eth_00233, SWS_Eth_91002, SWS_Eth_91003, SWS_Eth_91004, SWS_Eth_91005, SWS_Eth_91006

7 Functional specification

7.1 Ethernet BSW stack

As part of the AUTOSAR Layered Software Architecture according to Figure 7.1, the Ethernet BSW modules also form a layered software stack. Figure 7.1 depicts the basic structure of this Ethernet BSW stack. The Ethernet Interface module accesses several controllers using the Ethernet Driver layer, which can be made up of several Ethernet Drivers modules.

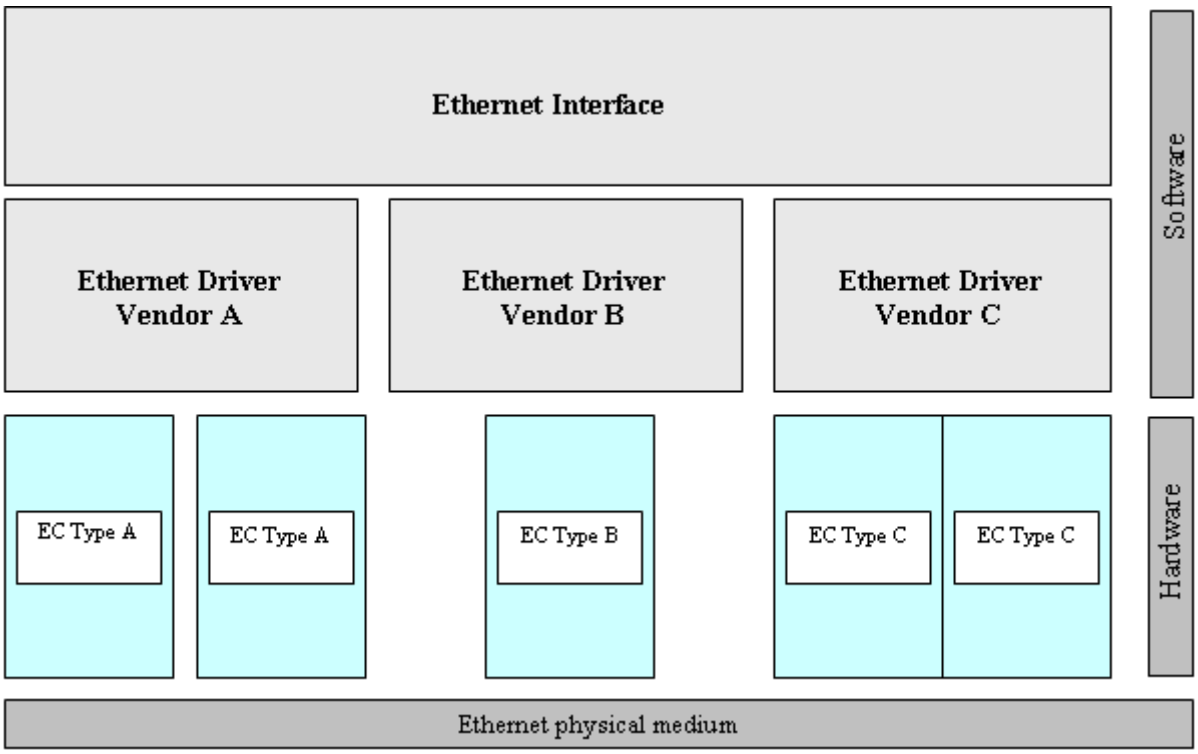

Figure 7.1: Basic Structure of the Ethernet BSW stack

Furthermore a Switch device might be connected to a dedicated controller index of an Ethernet Driver. This scenario leads to additional interaction between the Switch Driver and the Ethernet Driver [Figure 7.2]. The Ethernet Driver ask the Switch Driver for a special treatment to ensure that the current Ethernet frame could be managed in the Switch later on.

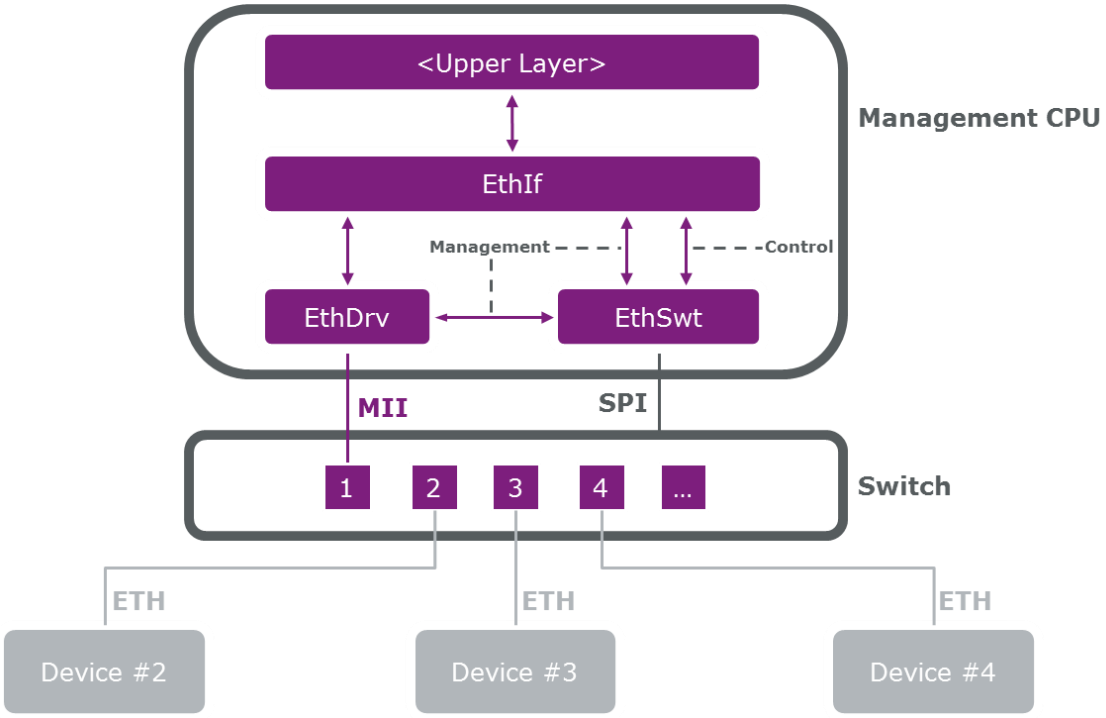

Figure 7.2: HW/SW basic structure including Switch device

7.1.1 Indexing scheme

Users of the Ethernet Driver identify controller resources using an indexing scheme as depicted in Figure 7.3.

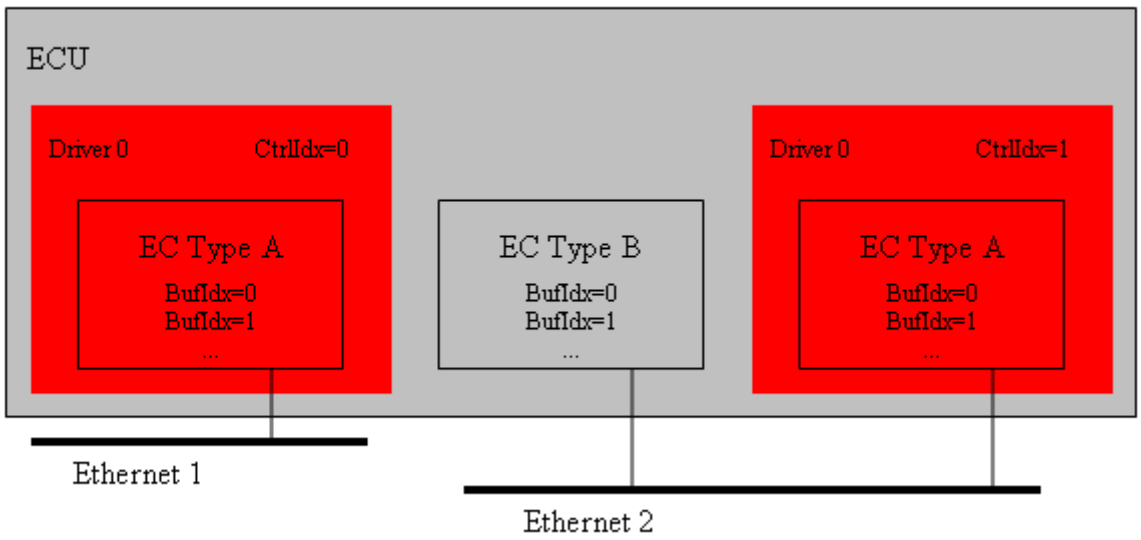

Figure 7.3: Ethernet Driver indexing scheme

[SWS_Eth_00003] [

The Ethernet Driver is using a zero-based index to abstract the access for upper software layers. The parameter Eth_CtrlIdx within configuration corresponds to parameter CtrlIdx used in the API. ⌋()

[SWS_Eth_00004] ⌈
A buffer index (BufIdx) indentifies an Ethernet buffer processed by Ethernet Driver API functions. Each controller's buffers are identified by buffer indexes 0 to (n-1) where n is the number of buffers processed by the corresponding controller. Buffer indexes are valid within a tuple <CtrlIdx, BufIdx> only. A BufIdx uniquely identifies the buffer used for an Ethernet Driver. ⌋()

7.1.2 Requirements

This chapter lists requirements that shall be fulfilled by Ethernet Driver module implementations.
The Ethernet Driver module environment comprises all modules which are calling interfaces of the Ethernet Driver module.

[SWS_Eth_00005] ⌈
The Ethernet Driver module shall support pre-compile time, link time and post-build time configuration. ⌋()

[SWS_Eth_00006] ⌈
The header file *Eth.h* shall include a software and specification version number. ⌋()

[SWS_Eth_00007] ⌈
The Ethernet Driver module shall perform a consistency check between code files and header files based on pre-process-checking the version numbers of related code files and header files. ⌋()

[SWS_Eth_00008] ⌈
In case development error detection is enabled for the Ethernet Driver module: The Ethernet Driver module shall check API parameters for validity and report detected errors to the DET. ⌋()

DET API functions are specified in [16].

[SWS_Eth_00011] ⌈
None of the Ethernet Driver module header files shall define global variables. ⌋()

[SWS_Eth_00218] ⌈
The Ethernet Driver shall ensure that the base addresses of all reception and transmission buffers fulfill the memory alignment requirements for all AUTOSAR data types of the respective platform. ⌋()

[SWS_Eth_00216] ⌈
For transmissions the Ethernet Controller shall enable hardware capabilities for the calculation of protocol checksums (offloading) according to the following list:

a) for IPv4 frames if EthCtrlEnableOffloadChecksumIPv4 is set to TRUE
b) for ICMP frames if EthCtrlEnableOffloadChecksumICMP is set to TRUE
c) for TCP frames if EthCtrlEnableOffloadChecksumTCP is set to TRUE
d) for UDP frames if EthCtrlEnableOffloadChecksumUDP is set to TRUE.
In all other cases, the Ethernet Controller shall not manipulate the checksum fields.]()

[SWS_Eth_00217] [
For reception the Ethernet Controller shall enable hardware capabilities to discard frames with mismatching protocol checksums (offloading) according to the following list:
a) for IPv4 frames if EthCtrlEnableOffloadChecksumIPv4 is set to TRUE
b) for ICMP frames if EthCtrlEnableOffloadChecksumICMP is set to TRUE
c) for TCP frames if EthCtrlEnableOffloadChecksumTCP is set to TRUE
d) for UDP frames if EthCtrlEnableOffloadChecksumUDP is set to TRUE.
In all other cases, the Ethernet Controller shall not consider the protocol checksum fields.]()

[SWS_Eth_00176] [
The Global Time interfaces shall be used to access the time synchronization functionalities (see document [23]).]()

[SWS_Eth_00243] [
Ethernet SW Driver shall call EthIf_TxConfirmation with Result set to E_OK to indicate a successful transmission; either from the Interrupt routine (in interrupt mode) or from the Eth_TxConfirmation routine in polling mode (if the notification has been enabled).] ()

[SWS_Eth_00256][
Ethernet SW Driver shall call EthIf_TxConfirmation with Result set to E_NOT_OK if the transmission failed.] ()

The call to EthIf_TxConfirmation with Result set to E_NOT_OK shall allow the upper layer to implement a simple locking scheme. It can rely on the fact that every time Eth_Transmit is called, EthIf_TxConfirmation will be called afterwards.

[SWS_Eth_00244] [
Ethernet SW Driver shall call EthIf_RxIndication to indicate a successful reception either from the Interrupt routine (in interrupt mode) or from the Eth_Receive routine in polling mode (please refer to SWC_ETH_0096)]()

[SWS_Eth_00247][
The Switch Driver management API's:

EthSwt_EthRxProcessFrame(),
EthSwt_EthRxFinishedIndication(),
EthSwt_EthTxPrepareFrame(),
EthSwt_EthTxAdaptBufferLength(),
EthSwt_EthTxProcessFrame() and
EthSwt_EthTxFinishedIndication()

shall be used to to inform the Switch Driver about a required special treatment for Switch management purpose (see document AUTOSAR_SWS_EthernetInterface).]()

7.1.3 Configuration description

[SWS_Eth_00012] [
The Ethernet Driver module shall provide an XML file that contains the data, which is required for the SW identification (it shall contain the vendor identification, module ID and software version information), configuration and integration process. This file should describe vendor specific configuration parameters as well as it should contain recommended configuration parameter values.]()

[SWS_Eth_00125] [
The MCG shall read the ECU configuration description of the Ethernet Driver module(s). Ethernet Driver related configuration data is contained in the Ethernet Driver module configuration description.]()

[SWS_Eth_00126] [
The MCG shall ensure the consistency of the generated configuration data.]()

[SWS_Eth_00013][
The configuration of the Ethernet Driver module shall be calculated at ECU configuration time. None of the communication parameters shall be calculated at runtime.]()

[SWS_Eth_00014] [
The start address of post-build time configuration data shall be passed during module initialization (see chapter 8.3.1).]()

An assignment of those configuration classes to configuration parameters can be found in chapter 10.

A detailed description of all Ethernet Driver related configuration parameters can be found in chapter 10 of this document.

7.2 Error classification

7.2.1 Development Errors

[SWS_Eth_00016] [

Type or error	*Relevance*	*Related error code*	*Value [hex]*
Invalid controller index	Development error	ETH_E_INV_CTRL_IDX	0x01
Eth module or	Development	ETH_E_UNINIT	0x02

controller was not initialized	error		
Invalid pointer in parameter list	Development error	ETH_E_PARAM_POINTER	0x03
Invalid parameter	Development error	ETH_E_INV_PARAM	0x04
Invalid mode	Development error	ETH_E_INV_MODE	0x05

]()

7.2.2 Runtime Errors

There are no runtime errors.

7.2.3 Transient Faults

There are no transient faults.

7.2.4 Production Errors

There are no production errors.

7.2.5 Extended Production Errors

Extended production errors are handled as events of the Diagnostic Event Manager. The event IDs are defined in the following tables, while the actual values are assigned externally by the configuration of the Diagnostic Event Manager, and are included in the module via Dem.h.

[SWS_Eth_00173] [

Error Name:	ETH_E_ACCESS	
Short Description:	Ethernet Controller Access Failure.	
Long Description:	Monitors the access to the Ethernet Controller.	
Detection Criteria:	Fail	When access to the Ethernet Controller fails the module shall report the extended production error with event status DEM_EVENT_STATUS_PREFAILED to DEM.
	Pass	When access to the Ethernet Controller succeds the module shall report the extended production error with event status DEM_EVENT_STATUS_PREPASSED to DEM.
Secondary Parameters:	None.	
Time Required:	None.	
Monitor Frequency	None.	

]()

[SWS_Eth_00174] [

Error Name:	ETH_E_RX_FRAMES_LOST	
Short Description:	Ethernet Frames Lost.	
Long Description:	Monitors the loss of Ethernet frames during reception.	
Detection Criteria:	Fail	When lost frames are detected the module shall report the

		extended production error with event status DEM_EVENT_STATUS_PREFAILED to DEM.
	Pass	When Ethernet Controller is successfully initialized the module shall report the extended production error with event status DEM_EVENT_STATUS_PREPASSED to DEM.
Secondary Parameters:	None.	
Time Required:	None.	
Monitor Frequency	None.	

]()
[SWS_Eth_00219] [

Error Name:	ETH_E_CRC	
Short Description:	CRC Failure	
Long Description:	Monitors invalid Ethernet frames during reception.	
Detection Criteria:	Fail	When invalid frames are detected the module shall report the extended production error with event status DEM_EVENT_STATUS_PREFAILED to DEM.
	Pass	When Ethernet Controller is successfully initialized the module shall report the extended production error with event status DEM_EVENT_STATUS_PREPASSED to DEM.
Secondary Parameters:	None.	
Time Required:	None.	
Monitor Frequency	None.	

]()
[SWS_Eth_00220] [

Error Name:	ETH_E_UNDERSIZEFRAME	
Short Description:	Frame Size Underflow	
Long Description:	Monitors undersize Ethernet frames during reception.	
Detection Criteria:	Fail	When invalid frames are detected the module shall report the extended production error with event status DEM_EVENT_STATUS_PREFAILED to DEM.
	Pass	When Ethernet Controller is successfully initialized the module shall report the extended production error with event status DEM_EVENT_STATUS_PREPASSED to DEM.
Secondary Parameters:	None.	
Time Required:	None.	
Monitor Frequency	None.	

]()
[SWS_Eth_00221] [

Error Name:	ETH_E_OVERSIZEFRAME	
Short Description:	Frame Size Overflow	
Long Description:	Monitors oversize Ethernet frames during reception.	
Detection Criteria:	Fail	When invalid frames are detected the module shall report the extended production error with event status DEM_EVENT_STATUS_PREFAILED to DEM.
	Pass	When Ethernet Controller is successfully initialized the module shall report the extended production error with event status DEM_EVENT_STATUS_PREPASSED to DEM.
Secondary Parameters:	None.	
Time Required:	None.	
Monitor Frequency	None.	

](
[SWS_Eth_00222] [

Error Name:	ETH_E_ALIGNMENT	
Short Description:	Frame Alignment Error	
Long Description:	Monitors alignment errors.	
Detection Criteria:	Fail	When invalid frames are detected the module shall report the extended production error with event status DEM_EVENT_STATUS_PREFAILED to DEM.
	Pass	When Ethernet Controller is successfully initialized the module shall report the extended production error with event status DEM_EVENT_STATUS_PREPASSED to DEM.
Secondary Parameters:	None.	
Time Required:	None.	
Monitor Frequency	None.	

](
[SWS_Eth_00223] [

Error Name:	ETH_E_SINGLECOLLISION	
Short Description:	Single Frame Collision	
Long Description:	Monitors Ethernet single frame collision.	
Detection Criteria:	Fail	When frame collisions are detected the module shall report the extended production error with event status DEM_EVENT_STATUS_PREFAILED to DEM.
	Pass	When Ethernet Controller is successfully initialized the module shall report the extended production error with event status DEM_EVENT_STATUS_PREPASSED to DEM.
Secondary Parameters:	None.	
Time Required:	None.	
Monitor Frequency	None.	

](
[SWS_Eth_00224] [

Error Name:	ETH_E_MULTIPLECOLLISION	
Short Description:	Multiple Frame Collision	
Long Description:	Monitors Ethernet multiple frame collision.	
Detection Criteria:	Fail	When fram collisions are detected the module shall report the extended production error with event status DEM_EVENT_STATUS_PREFAILED to DEM.
	Pass	When Ethernet Controller is successfully initialized the module shall report the extended production error with event status DEM_EVENT_STATUS_PREPASSED to DEM.
Secondary Parameters:	None.	
Time Required:	None.	
Monitor Frequency	None.	

](
[SWS_Eth_00225] [

Error Name:	ETH_E_LATECOLLISION	
Short Description:	Late Frame Collision	
Long Description:	Monitors Ethernet late frame collision.	
Detection Criteria:	Fail	When frame collisions are detected the module shall report the extended production error with event status DEM_EVENT_STATUS_PREFAILED to DEM.
	Pass	When Ethernet Controller is successfully initialized the

		module shall report the extended production error with event status DEM_EVENT_STATUS_PREPASSED to DEM.
Secondary Parameters:	None.	
Time Required:	None.	
Monitor Frequency	None.	

]()

8 API specification

8.1 Imported types

This chapter lists all types included from the following modules:

[SWS_Eth_00026] [

Module	*Header File*	*Imported Type*
ComStack_Types	ComStackTypes.h	BufReq_ReturnType
Dem	Rte_Dem_Type.h	Dem_EventIdType
	Rte_Dem_Type.h	Dem_EventStatusType
Eth_GeneralTypes	Eth_GeneralTypes.h	Eth_BufIdxType
	Eth_GeneralTypes.h	Eth_CounterType
	Eth_GeneralTypes.h	Eth_DataType
	Eth_GeneralTypes.h	Eth_FilterActionType
	Eth_GeneralTypes.h	Eth_FrameType
	Eth_GeneralTypes.h	Eth_ModeType
	Eth_GeneralTypes.h	Eth_RxStatsType
	Eth_GeneralTypes.h	Eth_RxStatusType
	Eth_GeneralTypes.h	Eth_TimeStampQualType
	Eth_GeneralTypes.h	Eth_TimeStampType
	Eth_GeneralTypes.h	Eth_TxErrorCounterValuesType
	Eth_GeneralTypes.h	Eth_TxStatsType
Std_Types	StandardTypes.h	Std_ReturnType
	StandardTypes.h	Std_VersionInfoType

] (SRS_Eth_00127)

8.2 Type definitions

8.2.1 Eth_ConfigType

[SWS_Eth_00156] [

Name:	`Eth_ConfigType`
Type:	`Structure`
Range:	Implementation specific.
Description:	Implementation specific structure of the post build configuration
Available via:	`Eth.h`

] ()

8.2.2 Eth_ModeType

[SWS_Eth_00158] [

Name:	`Eth_ModeType`		
Type:	`Enumeration`		
Range:	`ETH_MODE_DOWN`	0x00	Controller disabled
	`ETH_MODE_ACTIVE`	0x01	Controller enabled

Description:	This type defines the controller modes
Available via:	`Eth_GeneralTypes.h`

] ()

8.2.3 Eth_StateType

[SWS_Eth_00159] [

Name:	`Eth_StateType`		
Type:	`Enumeration`		
Range:	`ETH_STATE_UNINIT`	0x00	Driver is not yet configured
	`ETH_STATE_INIT`	0x01	Driver is configured
Description:	Status supervision used for Development Error Detection. The state shall be available for debugging.		
Available via:	`Eth_GeneralTypes.h`		

] ()

8.2.4 Eth_FrameType

[SWS_Eth_00160] [

Name:	`Eth_FrameType`
Type:	`uint16`
Description:	This type defines the Ethernet frame type used in the Ethernet frame header
Available via:	`Eth_GeneralTypes.h`

] ()

8.2.5 Eth_DataType

[SWS_Eth_00161] [

Name:	`Eth_DataType`
Type:	`uint8, uint16, uint32`
Description:	This type defines the Ethernet data type used for data transmission. Its definition depends on the used CPU.
Available via:	`Eth_GeneralTypes.h`

] ()

8.2.6 Eth_BufIdxType

[SWS_Eth_00175] [

Name:	`Eth_BufIdxType`
Type:	`uint32`
Description:	Ethernet buffer identifier type.
Available via:	`Eth_GeneralTypes.h`

] ()

8.2.7 Eth_RxStatusType

[SWS_Eth_00162] [

Name:	`Eth_RxStatusType`

Type:	Enumeration		
Range:	ETH_RECEIVED	0x00	Ethernet frame has been received, no further frames available
	ETH_NOT_RECEIVED	0x01	Ethernet frame has not been received, no further frames available
	ETH_RECEIVED_MORE_DATA_AVAILABLE	0x02	Ethernet frame has been received, more frames are available
Description:	Used as out parameter in Eth_Receive() indicates whether a frame has been received and if so, whether more frames are available or frames got lost.		
Available via:	Eth_GeneralTypes.h		

] ()

8.2.8 Eth_FilterActionType

[SWS_Eth_00163] [

Name:	Eth_FilterActionType		
Type:	Enumeration		
Range:	ETH_ADD_TO_FILTER	0x00	add the MAC address to the filter, meaning allow reception
	ETH_REMOVE_FROM_FILTER	0x01	remove the MAC address from the filter, meaning reception is blocked in the lower layer
Description:	The Enumeration Type Eth_FilterActionType describes the action to be taklen for the MAC address given in *PhysAddrPtr.		
Available via:	Eth_GeneralTypes.h		

] ()

8.2.9 Eth_TimeStampQualType

[SWS_Eth_00177] [

Name:	Eth_TimeStampQualType		
Type:	--		
Range:	ETH_VALID	0	--
	ETH_INVALID	1	--
	ETH_UNCERTAIN	2	--
Description:	Depending on the HW, quality information regarding the evaluated time stamp might be supported. If not supported, the value shall be always Valid. For Uncertain and Invalid values, the upper layer shall discard the time stamp.		
Available via:	Eth_GeneralTypes.h		

] ()

8.2.10 Eth_TimeStampType

[SWS_Eth_00178] [

Name:	Eth_TimeStampType		
Type:	Structure		
Element:	uint32	nanoseconds	Nanoseconds part of the time
	uint32	seconds	32 bit LSB of the 48 bits Seconds part of the time

	uint16	secondsHi	16 bit MSB of the 48 bits Seconds part of the time
Description:	Variables of this type are used for expressing time stamps including relative time and absolute calendar time. The absolute time starts at 1970-01-01. 0 to 281474976710655s == 3257812230d [0xFFFF FFFF FFFF] 0 to 999999999ns [0x3B9A C9FF] invalid value in nanoseconds: [0x3B9A CA00] to [0x3FFF FFFF] Bit 30 and 31 reserved, default: 0		
Available via:	Eth_GeneralTypes.h		

] ()

8.2.11 Eth_TimeIntDiffType

[SWS_Eth_00179] [

Name:	Eth_TimeIntDiffType		
Type:	Structure		
Element:	Eth_TimeStampType	diff	time difference
	boolean	sign	Positive (True) / negative (False) time
Description:	Variables of this type are used to express time differences.		
Available via:	Eth_GeneralTypes.h		

] ()

8.2.12 Eth_RateRatioType

[SWS_Eth_00180] [

Name:	Eth_RateRatioType		
Type:	Structure		
Element:	Eth_TimeIntDiffType	IngressTimeStampDelta	IngressTimeStampSync2 - IngressTimeStampSync1
	Eth_TimeIntDiffType	OriginTimeStampDelta	OriginTimeStampSync2[FUP2] - OriginTimeStampSync1[FUP1]
Description:	Variables of this type are used to express frequency ratios.		
Available via:	Eth_GeneralTypes.h		

] ()

8.2.13 Eth_MacVlanType

[SWS_Eth_91001] [

Name:	Eth_MacVlanType		
Type:	Structure		
Element:	uint8[6]	MacAddr	Specifies the MAC address [0..255,0..255,0..255,0..255,0..255,0..255]
	uint16	VlanId	Specifies the VLAN address 0..65535
	uint32	SwitchPort	Specifies the ports of the switch as bit mask (0x00000001->Port0, 0x80000001-

			>Port31+Port0)
Description:	This type is used to read out addresses from the address resolution logic (ARL) table of the switch. typedef struct { uint8 MacAddr[6U]; uint16 VlanId; uint32 SwitchPort; } Eth_MacVlanType; In case of Macaddr contains a Multicast Address MacVlanType.SwitchPort shall be handled as Bitmask, each bit represents a Switch Port, Bit 0 represents EthSwichtPortIdx = 0 , Bit 1 represents EthSwichtPortIdx = 1 and so on. In case of Macaddr contains not a Multicast Address MacVlanType.SwitchPort shall be handled as a value representing the EthSwitchPortIdx.		
Available via:	`Eth_GeneralTypes.h`		

] (SRS_ETH_00086)

8.2.14 Eth_CounterType

[SWS_Eth_91007] [

Name:	`Eth_CounterType`		
Type:	`Structure`		
Element:	`uint32`	`DropPktBufOverrun`	dropped packets due to buffer overrun
	`uint32`	`DropPktCrc`	dropped packets due to CRC errors
	`uint32`	`UndersizePkt`	number of undersize packets which were less than 64 octets long (excluding framing bits, but including FCS octets) and were otherwise well formed. (see IETF RFC 1757)
	`uint32`	`OversizePkt`	number of oversize packets which are longer than 1518 octets (excluding framing bits, but including FCS octets) and were otherwise well formed. (see IETF RFC 1757)
	`uint32`	`AlgnmtErr`	number of alignment errors, i.e. packets which are received and are not an integral number of octets in length and do not pass the CRC.
	`uint32`	`SqeTestErr`	SQE test error according to IETF RFC1643 dot3StatsSQETestErrors
	`uint32`	`DiscInbdPkt`	The number of inbound packets which were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol. One possible reason for discarding such a packet could be to free up buffer space. (see IETF RFC 2233 ifInDiscards)
	`uint32`	`ErrInbdPkt`	total number of erroneous inbound packets
	`uint32`	`DiscOtbdPkt`	The number of outbound packets which were chosen to be discarded even though no errors had been detected to prevent their being transmitted. One possible reason for discarding such a packet could be to free up buffer space.

			(see IETF RFC 2233 ifOutDiscards)
	uint32	ErrOtbdPkt	total number of erroneous outbound packets
	uint32	SnglCollPkt	Single collision frames: A count of successfully transmitted frames on a particular interface for which transmission is inhibited by exactly one collision. (see IETF RFC1643 dot3StatsSingleCollisionFrames)
	uint32	MultCollPkt	Multiple collision frames: A count of successfully transmitted frames on a particular interface for which transmission is inhibited by more than one collision. (see IETF RFC1643 dot3StatsMultipleCollisionFrames)
	uint32	DfrdPkt	Number of deferred transmission: A count of frames for which the first transmission attempt on a particular interface is delayed because the medium is busy. (see IETF RFC1643 dot3StatsDeferredTransmissions)
	uint32	LatCollPkt	Number of late collisions: The number of times that a collision is detected on a particular interface later than 512 bit-times into the transmission of a packet. (see IETF RFC1643 dot3StatsLateCollisions)
	uint32	HwDepCtr0	hardware dependent counter value
	uint32	HwDepCtr1	hardware dependent counter value
	uint32	HwDepCtr2	hardware dependent counter value
	uint32	HwDepCtr3	hardware dependent counter value
Description:	Statistic counter for diagnostics.		
Available via:	Eth_GeneralTypes.h		

] ()

8.2.15 Eth_RxStatsType

[SWS_Eth_91002] [

Name:	Eth_RxStatsType		
Type:	Structure		
Element:	uint32	RxStatsDropEvents	The total number of events in which packets were dropped by the probe due to lack of resources. Also described in IETF RFC 2819 MIB etherStatsDropEvents.
	uint32	RxStatsOctets	The total number of octets of data (including those in bad packets) received on the network (excluding framing bits but including FCS octets). Also described in IETF RFC 2819 MIB etherStatsOctets.
	uint32	RxStatsPkts	The total number of packets (including bad packets, broadcast packets, and multicast packets) received. Also described in IETF

			RFC 2819 MIB etherStatsPkts
	uint32	RxStatsBroadcastPkts	The total number of good packets received that were directed to the broadcast address. Also described in IETF RFC 2819 MIB etherStatsBroadcastPkts
	uint32	RxStatsMulticastPkts	The total number of good packets received that were directed to a multicast address. Also described in IETF RFC 2819 MIB etherStatsMulticastPkts.
	uint32	RxStatsCrcAlignErrors	The total number of packets received that had a length of bertween 64 and 1518 octets that had either a bad Frame Check Sequence (FCS) with an integral number of octets (FCS Error) or a bad FCS with a non-integral number of octets (Alignment Error). Also described in IETF RFC 2819 MIB etherStatsCRCAlignErrors
	uint32	RxStatsUndersizePkts	The total number of packets received that were less than 64 octets long (excluding framing bits, but including FCS octets) and were otherwise well formed. Also described in IETF RFC 2819 MIB etherStatsUndersizePkts.
	uint32	RxStatsOversizePkts	The total number of packets received that were longer than 1518 octets (excluding framing bits, but including FCS octets) and were otherwise well formed. Also described in IETF RFC 2819 MIB etherStatsOversizePkts
	uint32	RxStatsFragments	The total number of packets received that were less than 64 octets in length (excluding framing bits but including FCS octets) and had either a bad Frame Check Sequence (FCS) with an integral number of octets (FCS Error) or a bad FCS with a non-integral number of octets (Alignment Error). Also described in IETF RFC 2819 MIB etherStatsFragments.
	uint32	RxStatsJabbers	The total number of packets received that were longer than 1518 octets, and had either a bad Frame Check Sequence (FCS) with an integral number of octets (FCS Error) or a bad FCS with a non-integral number of octets (Alignment Error). Also described in IETF RFC 2819 MIB etherStatsJabbers.
	uint32	RxStatsCollisions	The best estimate of the total number of collisions on this Ethernet segment. Also described

			in IETF RFC 2819 MIB etherStatsCollisions
	uint32	RxStatsPkts64Octets	The total number of packets (including bad packets) received that were 64 octets in length. Also described in IETF RFC 2819 MIB etherStatsPkts64Octets
	uint32	RxStatsPkts65to127Octets	The total number of packets (including bad packets) received that were between 65 and 127 octets in length. Also described in IETF RFC 2819 MIB etherStatsPkts65to127Octets
	uint32	RxStatsPkts128to255Octets	The total number of packets (including bad packets) received that were between 128 and 255 octets in length. Also described in IETF RFC 2819 MIB etherStatsPkts128to255Octets
	uint32	RxStatsPkts256to511Octets	The total number of packets (including bad packets) received that were between 256 and 511 octets in length. Also described in IETF RFC 2819 MIB etherStatsPkts256to511Octets
	uint32	RxStatsPkts512to1023Octets	The total number of packets (including bad packets) received that were between 512 and 1023 octets in length. Also described in IETF RFC 2819 MIB etherStatsPkts512to1023Octets
	uint32	RxStatsPkts1024to1518Octets	The total number of packets (including bad packets) received that were between 1024 and 1518 octets in length. Also described in IETF RFC 2819 MIB etherStatsPkts1024to1518Octets
	uint32	RxUnicastFrames	The number of subnetwork-unicast packets delivered to a higher-layer protocol. Also described in IETF RFC1213 MIB ifInUcastPkts
Description:	Statistic counter for diagnostics.		
Available via:	Eth_GeneralTypes.h		

] (SRS_Eth_00127)

8.2.16 Eth_TxStatsType

[SWS_Eth_91003] [

Name:	Eth_TxStatsType		
Type:	Structure		
Element:	uint32	TxNumberOfOctets	The total number of octets transmitted out of the interface, including framing characters. Also described in IETF RFC1213 MIB ifOutOctets.
	uint32	TxNUcastPkts	The total number of packets that higher-

			level protocols requested be transmitted to a non-unicast (i.e., a subnetwork-broadcast or subnetwork-multicast) address, including those that were discarded or not sent. Also described in IETF RFC1213 MIB ifOutNUcastPkts
	uint32	TxUniCastPkts	The total number of packets that higher-level protocols requested be transmitted to a subnetwork-unicast address, including those that were discarded or not sent. Also described in IETF RFC1213 MIB ifOutUcastPkts.
Description:	Statistic counter for diagnostics.		
Available via:	Eth_GeneralTypes.h		

] (SRS_Eth_00127)

8.2.17 Eth_TxErrorCounterValuesType

[SWS_Eth_91004] [

Name:	Eth_TxErrorCounterValuesType		
Type:	Structure		
Element:	uint32	TxDroppedNoErrorPkts	The number of outbound packets which were chosen to be discarded even though no errors had been detected to prevent their being transmitted. One possible reason for discarding such a packet could be to free up buffer space. Also described in IETF RFC1213 MIB ifOutDiscards
	uint32	TxDroppedErrorPkts	transmitted because of errors. Also described in IETF RFC1213 MIB ifOutErrors
	uint32	TxDeferredTrans	A count of frames for which the first transmission attempt on a particular interface is delayed because the medium is busy. The count represented by an instance of this object does not include frames involved in collisions. Also described in IETF RFC1643 MIB dot3StatsDeferredTransmissions
	uint32	TxSingleCollision	A count of successfully transmitted frames on a particular interface for which transmission is inhibited by exactly one collision. A frame that is counted by an instance of this object is also counted by the corresponding instance of either the TxUniCastPkts and TxNUcastPkts and is not counted by the corresponding instance of the TxMultipleCollision object. Also described in IETF RFC1643 MIB dot3StatsSingleCollisionFrames
	uint32	TxMultipleCollision	A count of successfully transmitted frames on a particular interface for which transmission is inhibited by

			more than one collision. A frame that is counted by an instance of this object is also counted by the corresponding instance of either the TxUniCastPkts and TxNUcastPkts and is not counted by the corresponding instance of the TxSingleCollision object. Also described in IETF RFC1643 MIB dot3StatsMultipleCollisionFrames.
	uint32	TxLateCollision	The number of times that a collision is detected on a particular interface later than 512 bit-times into the transmission of a packet. Five hundred and twelve bit-times corresponds to 51.2 microseconds on a 10 Mbit/s system. A (late) collision included in a count represented by an instance of this object is also considered as a (generic) collision for purposes of other collision-related statistics. Also described in IETF RFC1643 MIB dot3StatsLateCollisions
	uint32	TxExcessiveCollison	A count of frames for which transmission on a particular interface fails due to excessive collisions. Also described in IETF RFC1643 MIB dot3StatsExcessiveCollisions
Description:	Statistic counters for diagnostics.		
Available via:	Eth_GeneralTypes.h		

] (SRS_Eth_00127)

8.3 Function definitions

This is a list of functions provided for upper layer modules.

8.3.1 Eth_Init

[SWS_Eth_00027] [

Service name:	Eth_Init	
Syntax:	void Eth_Init(const Eth_ConfigType* CfgPtr)	
Service ID[hex]:	0x01	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CfgPtr	Points to the implementation specific structure
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Initializes the Ethernet Driver	
Available via:	Eth.h	

] () [SWS_Eth_00028][

The function shall store the access to the configuration structure for subsequent API calls.]()

[SWS_Eth_00034] [
The function shall for all configured Ethernet controllers in the current EthConfigSet:

- Disable all controller
- Clear pending Ethernet interrupts
- Configure all controller configuration parameters (e.g. interrupts, frame length, frame filter, ...)
- Configure all transmit / receive resources (e.g. buffer initialization)
- delete all pending transmit and receive requests]()

[SWS_Eth_00029][
The function shall change the state of the component from ETH_STATE_UNINIT to ETH_STATE_INIT.]()

[SWS_Eth_00039] [
The function shall check the access to the Ethernet controller. If the check fails, the function shall raise the production error ETH_E_ACCESS.]()

[SWS_Eth_00031][
Caveat: The API has to be called during initialization.]()

8.3.2 Eth_SetControllerMode

[SWS_Eth_00041] [

Service name:	Eth_SetControllerMode	
Syntax:	`Std_ReturnType Eth_SetControllerMode( uint8 CtrlIdx, Eth_ModeType CtrlMode )`	
Service ID[hex]:	0x03	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the controller within the context of the Ethernet Driver
	CtrlMode	ETH_MODE_DOWN: disable the controller ETH_MODE_ACTIVE: enable the controller
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: controller mode could not be changed
Description:	Enables / disables the indexed controller	
Available via:	`Eth.h`	

] () [SWS_Eth_00042] [
The function shall:

- Put the controller in the specified mode given in the parameter ‘CtrlMode’
  - Upon mode ETH_MODE_DOWN the driver shall:
    - Disable the Ethernet controller
    - Reset all transmit and receive buffers (i.e. ignore all pending transmission and reception requests)

- Upon mode ETH_MODE_ACTIVE:
  - Enable all transmit and receive buffers
  - Enable the Ethernet controller⌋()

[SWS_Eth_00043] ⌈
If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT otherwise (if DET is disabled) return E_NOT_OK. ⌋()

[SWS_Eth_00044] ⌈
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX otherwise (if DET is disabled) return E_NOT_OK. ⌋()

[SWS_Eth_00168] ⌈
The function shall check the access to the Ethernet controller. If the check fails, the function shall raise the production error ETH_E_ACCESS and return E_NOT_OK.⌋()

[SWS_Eth_00045] ⌈
Caveat: The function requires previous controller initialization (Eth_Init). ⌋()

8.3.3 Eth_GetControllerMode

[SWS_Eth_00046] ⌈

Service name:	Eth_GetControllerMode	
Syntax:	`Std_ReturnType Eth_GetControllerMode( uint8 CtrlIdx, Eth_ModeType* CtrlModePtr )`	
Service ID[hex]:	0x04	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the controller within the context of the Ethernet Driver
Parameters (inout):	None	
Parameters (out):	CtrlModePtr	ETH_MODE_DOWN: the controller is disabled ETH_MODE_ACTIVE: the controller is enabled
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: controller mode could not be obtained
Description:	Obtains the state of the indexed controller	
Available via:	`Eth.h`	

⌋ () [SWS_Eth_00047] ⌈
The function shall read the current controller mode. ⌋()

[SWS_Eth_00048] ⌈
If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT otherwise (if DET is disabled) return E_NOT_OK. ⌋()

[SWS_Eth_00049] ⌈
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX otherwise (if DET is disabled) return E_NOT_OK. ⌋()

[SWS_Eth_00050] ⌈
If development error detection is enabled: the function shall check the parameter CtrlModePtr for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER otherwise (if DET is disabled) return E_NOT_OK. ⌋()

[SWS_Eth_00051] ⌈
Caveat: The function requires previous controller initialization (Eth_Init). ⌋()

8.3.4 Eth_GetPhysAddr

[SWS_Eth_00052] ⌈

Service name:	Eth_GetPhysAddr	

Syntax:	`void Eth_GetPhysAddr(`
	`  uint8 CtrlIdx,`
	`  uint8* PhysAddrPtr`
	`)`	

Service ID[hex]:	0x08	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the controller within the context of the Ethernet Driver
Parameters (inout):	None	
Parameters (out):	PhysAddrPtr	Physical source address (MAC address) in network byte order.
Return value:	void	None
Description:	Obtains the physical source address used by the indexed controller	
Available via:	`Eth.h`	

⌋ ()
[SWS_Eth_00053] ⌈
The function shall read the source address used by the indexed controller. ⌋()

[SWS_Eth_00054] ⌈
If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT. ⌋()

[SWS_Eth_00055] ⌈
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX. ⌋()

[SWS_Eth_00056] ⌈
If development error detection is enabled: the function shall check the parameter PhysAddrPtr for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER. ⌋()

[SWS_Eth_00057] ⌈
Caveat: The function requires previous controller initialization (Eth_Init). ⌋()

8.3.5 Eth_SetPhysAddr

[SWS_Eth_00151] ⌈

Service name:	Eth_SetPhysAddr	
Syntax:	`void Eth_SetPhysAddr( uint8 CtrlIdx, const uint8* PhysAddrPtr )`	
Service ID[hex]:	0x13	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same CtrlIdx, reentrant for different	
Parameters (in):	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Driver.
	PhysAddrPtr	Pointer to memory containing the physical source address (MAC address) in network byte order.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Sets the physical source address used by the indexed controller	
Available via:	`Eth.h`	

⌋ () [SWS_Eth_00139] ⌈
The function shall update the source address used by the indexed controller. ⌋()

[SWS_Eth_00140] ⌈
If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT. ⌋()

[SWS_Eth_00141] ⌈
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX. ⌋()

[SWS_Eth_00142] ⌈
If development error detection is enabled: the function shall check the parameter PhysAddrPtr for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER. ⌋()

[SWS_Eth_00143] ⌈
Caveat: The function requires previous controller initialization (Eth_Init). ⌋()

8.3.6 Eth_UpdatePhysAddrFilter

[SWS_Eth_00152] ⌈

Service name:	Eth_UpdatePhysAddrFilter
Syntax:	`Std_ReturnType Eth_UpdatePhysAddrFilter( uint8 CtrlIdx,`

	`const uint8* PhysAddrPtr,`
	`Eth_FilterActionType Action`
	`)`	

Service ID[hex]:	0x12	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same CtrlIdx, reentrant for different	
Parameters (in):	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Driver
	PhysAddrPtr	Pointer to memory containing the physical destination address (MAC address) in network byte order. This is the multicast destination address of the layer 2 Ethernet packet.
	Action	Add or remove the address from the Ethernet controllers filter.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: filter was successfully changed E_NOT_OK: filter could not be changed
Description:	Update the physical source address to/from the indexed controller filter. If the Ethernet Controller is not capable to do the filtering, the software has to do this.	
Available via:	`Eth.h`	

] () [SWS_Eth_00150] [
The function shall update the physical address receive filter of the indexed controller.]()

[SWS_Eth_00245][
The Ethernet driver module will receive a frame when the destination Address match the PhyAddrPtr passed here. (e.g matching can be done via hash table or simple pattern matching)] ()

Note: Underlying HW mechanism can be used if available. Otherwise the Ethernet driver needs to do this by software.

[SWS_Eth_00246][
If the matching is positive, the upper layer shall be notified by calling RxIndication() callback.
If the matching is negative, the frame shall be discarded.]()

[SWS_Eth_00164] [
If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT.]()

[SWS_Eth_00165] [
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.]()

[SWS_Eth_00166] [
If development error detection is enabled the function shall check the parameter PhysAddrPtr for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER.]()

[SWS_Eth_00167] ⌈
Caveat: The function requires previous controller initialization (Eth_Init). ⌋()

[SWS_Eth_00144] ⌈
If the physical source address (MAC address) is set to FF:FF:FF: FF:FF:FF, this shall completely open the filter. ⌋()

[SWS_Eth_00146] ⌈
If this API is used and the hardware does not support filtering, promiscuous mode shall be enabled during initialization. ⌋()

[SWS_Eth_00147] ⌈
If the physical source address (MAC address) is set to 00:00:00: 00:00:00, this shall reduce the filter to the controllers unique unicast MAC address and end promiscuous mode if it was turned on. ⌋()

8.3.7 Eth_WriteMii

[SWS_Eth_00058] ⌈

Service name:	Eth_WriteMii	

Syntax:	`Std_ReturnType Eth_WriteMii(`
	`  uint8 CtrlIdx,`
	`  uint8 TrcvIdx,`
	`  uint8 RegIdx,`
	`  uint16 RegVal`
	`)`	

Service ID[hex]:	0x05	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the controller within the context of the Ethernet Driver
	TrcvIdx	Index of the transceiver on the MII (see [21] for details)
	RegIdx	Index of the transceiver register on the MII (see [21] for details)
	RegVal	Value to be written into the indexed register (see [21] for details)
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Service accepted E_NOT_OK: Service denied
Description:	Configures a transceiver register or triggers a function offered by the receiver	
Available via:	`Eth.h`	

⌋ () [SWS_Eth_00059] ⌈
The function shall write the specified transceiver register through the MII of the indexed controller. ⌋()

[SWS_Eth_00241]⌈
The function shall call EthTrcv_WriteMiiIndication when the MII access finished.⌋ ()

[SWS_Eth_00060] ⌈
If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT. ⌋()

[SWS_Eth_00061] ⌈
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX. ⌋()

[SWS_Eth_00062] ⌈
The function shall be pre compile time configurable On/Off by the configuration parameter: EthCtrlEnableMii. ⌋()

[SWS_Eth_00063] ⌈
Caveat: The function requires previous controller initialization (Eth_Init). ⌋()

8.3.8 Eth_ReadMii

[SWS_Eth_00064] ⌈

Service name:	Eth_ReadMii	

Syntax:	`Std_ReturnType Eth_ReadMii(`
	`  uint8 CtrlIdx,`
	`  uint8 TrcvIdx,`
	`  uint8 RegIdx,`
	`  uint16* RegValPtr`
	`)`	

Service ID[hex]:	0x06	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the controller within the context of the Ethernet Driver
	TrcvIdx	Index of the transceiver on the MII (see [21] for details)
	RegIdx	Index of the transceiver register on the MII (see [21] for details)
Parameters (inout):	None	
Parameters (out):	RegValPtr	Filled with the register content of the indexed register (see [21] for details)
Return value:	Std_ReturnType	E_OK: Service accepted E_NOT_OK: Service denied
Description:	Reads a transceiver register	
Available via:	`Eth.h`	

⌋ () [SWS_Eth_00065] ⌈
The function shall read the specified transceiver register through the MII of the indexed controller. ⌋()

[SWS_Eth_00242]⌈
The function shall call EthTrcv_ReadMiiIndication when the MII access finished.⌋ ()
[SWS_Eth_00066] ⌈
If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT. ⌋()

[SWS_Eth_00067] ⌈

If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.]()

[SWS_Eth_00068] ⌈
If development error detection is enabled: the function shall check the parameter RegValPtr for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER.]()

[SWS_Eth_00069] ⌈
The function shall be pre compile time configurable On/Off by the configuration parameter: EthCtrlEnableMii.]()

[SWS_Eth_00070] ⌈
Caveat: The function requires previous controller initialization (Eth_Init).]()

8.3.9 Eth_GetCounterValues

[SWS_Eth_00226] ⌈

Service name:	Eth_GetCounterValues	

Syntax:	`Std_ReturnType Eth_GetCounterValues(`
	`  uint8 CtrlIdx,`
	`  Eth_CounterType* CounterPtr`
	`)`	

Service ID[hex]:	0x14	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the controller within the context of the Ethernet Driver
Parameters (inout):	None	
Parameters (out):	CounterPtr	counter values according to IETF RFC 1757, RFC 1643 and RFC 2233.
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: counter values read failure
Description:	Reads a list with drop counter values of the corresponding controller. The meaning of these values is described at Eth_CounterType.	
Available via:	`Eth.h`	

⌋ (SRS_Eth_00127) [SWS_Eth_00227] ⌈
The function shall read a list of values from the indexed controller.]()

[SWS_Eth_00228] ⌈
If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT otherwise (if DET is disabled) return E_NOT_OK.]()

[SWS_Eth_00229] ⌈
If dev development elopment error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the

development error ETH_E_INV_CTRL_IDX otherwise (if DET is disabled) return E_NOT_OK.](SWS_Eth_00230)

[SWS_Eth_00230] [
If development error detection is enabled: the function shall check the parameter CounterPtr for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER otherwise (if DET is disabled) return E_NOT_OK.](SWS_Eth_00231)

[SWS_Eth_00231] [
The function Eth_GetCounterValues shall be pre compile time configurable On/Off by the configuration parameter: EthGetCounterValuesApi.](SWS_Eth_00232)

[SWS_Eth_00232] [
Caveat: The function requires previous controller initialization (Eth_Init).](SWS_Eth_00233)

8.3.10 Eth_GetRxStats

[SWS_Eth_00233] [

Service name:	Eth_GetRxStats	

Syntax:	`Std_ReturnType Eth_GetRxStats(`
	`  uint8 CtrlIdx,`
	`  Eth_RxStatsType* RxStats`
	`)`	

Service ID[hex]:	0x15	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the controller within the context of the Ethernet Driver
Parameters (inout):	None	
Parameters (out):	RxStats	List of values according to IETF RFC 2819 (Remote Network Monitoring Management Information Base)
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: drop counter could not be obtained
Description:	Returns the following list according to IETF RFC2819, where the maximal possible value shall denote an invalid value, e.g. if this counter is not available: 1. etherStatsDropEvents 2. etherStatsOctets 3. etherStatsPkts 4. etherStatsBroadcastPkts 5. etherStatsMulticastPkts 6. etherStatsCrcAlignErrors 7. etherStatsUndersizePkts 8. etherStatsOversizePkts 9. etherStatsFragments 10. etherStatsJabbers 11. etherStatsCollisions 12. etherStatsPkts64Octets 13. etherStatsPkts65to127Octets 14. etherStatsPkts128to255Octets 15. etherStatsPkts256to511Octets 16. etherStatsPkts512to1023Octets 17. etherStatsPkts1024to1518Octets	

Available via:	`Eth.h`

] (SRS_Eth_00127) [SWS_Eth_00234] ⌈
The function shall read a list of values from the indexed controller according to [22].
⌋()

[SWS_Eth_00235] ⌈
If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT otherwise (if DET is disabled) return E_NOT_OK.
⌋()

[SWS_Eth_00236] ⌈
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX otherwise (if DET is disabled) return E_NOT_OK. ⌋()

[SWS_Eth_00237] ⌈
If development error detection is enabled: the function shall check the parameter RxStats for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER otherwise (if DET is disabled) return E_NOT_OK.
⌋()

[SWS_Eth_00238] ⌈
The function Eth_GetRxStats shall be pre compile time configurable On/Off by the configuration parameter: EthGetRxStatsApi. ⌋()

8.3.11 Eth_GetTxStats

[SWS_Eth_91005] ⌈

Service name:	Eth_GetTxStats	

Syntax:	`Std_ReturnType Eth_GetTxStats(`
	`  uint8 CtrlIdx,`
	`  Eth_TxStatsType* TxStats`
	`)`	

Service ID[hex]:	0x1c	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the controller within the context of the Ethernet Driver
Parameters (inout):	None	
Parameters (out):	TxStats	List of values to read statistic values for transmission.
Return value:	Std_ReturnType	E_OK: success, E_NOTOK: Tx-statistics could not be obtained
Description:	Returns the list of Transmission Statistics out of IETF RFC1213 defined with Eth_TxStatsType, where the maximal possible value shall denote an invalid value, e.g. this counter is not available.	
Available via:	`Eth.h`	

] (SRS_Eth_00127)

[SWS_Eth_00248][
If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT otherwise (if DET is disabled) return E_NOT_OK.] (SRS_BSW_00101, SRS_BSW_00416)

[SWS_Eth_00249][
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX otherwise (if DET is disabled) return E_NOT_OK.] (SRS_BSW_00323, SRS_BSW_00369)

[SWS_Eth_00250][
If development error detection is enabled: the function shall check the parameter TxStats for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER otherwise (if DET is disabled) return E_NOT_OK.] (SRS_BSW_00323, SRS_BSW_00369)

[SWS_Eth_00251][
The function Eth_GetTxStats shall be pre compile time configurable On/Off by the configuration parameter: EthGetTxStatsApi.] (SRS_Eth_00053)

8.3.12 Eth_GetTxErrorCounterValues

[SWS_Eth_91006] [

Service name:	Eth_GetTxErrorCounterValues	
Syntax:	`Std_ReturnType Eth_GetTxErrorCounterValues( uint8 CtrlIdx, Eth_TxErrorCounterValuesType* TxErrorCounterValues )`	
Service ID[hex]:	0x1d	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the controller within the context of the Ethernet Driver
Parameters (inout):	None	
Parameters (out):	TxErrorCounterValues	List of values to read statistic error counter values for transmission.
Return value:	Std_ReturnType	E_OK: success, E_NOTOK: Tx-statistics could not be obtained
Description:	Returns the list of Transmission Error Counters out of IETF RFC1213 and RFC1643 defined with Eth_TxErrorCounterValuesType, where the maximal possible value shall denote an invalid value, e.g. this counter is not available.	
Available via:	`Eth.h`	

] (SRS_Eth_00127)

[SWS_Eth_00252][
If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the

development error ETH_E_UNINIT otherwise (if DET is disabled) return E_NOT_OK.] (SRS_BSW_00101, SRS_BSW_00416)

[SWS_Eth_00253][
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX otherwise (if DET is disabled) return E_NOT_OK.] (SRS_BSW_00323, SRS_BSW_00369)

[SWS_Eth_00254][
If development error detection is enabled: the function shall check the parameter TxStats for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER otherwise (if DET is disabled) return E_NOT_OK.] (SRS_BSW_00323, SRS_BSW_00369)

[SWS_Eth_00255][
The function Eth_GetTxErrorCounterValues shall be pre compile time configurable On/Off by the configuration parameter: EthGetTxErrorCounterValuesApi.] (SRS_Eth_00053)

8.3.13 Eth_GetCurrentTime

[SWS_Eth_00181] [

Service name:	Eth_GetCurrentTime	

Syntax:	`Std_ReturnType Eth_GetCurrentTime(`
	`  uint8 CtrlIdx,`
	`  Eth_TimeStampQualType* timeQualPtr,`
	`  Eth_TimeStampType* timeStampPtr`
	`)`	

Service ID[hex]:	0x16	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the addresses ETH controller.
Parameters (inout):	None	
Parameters (out):	timeQualPtr	quality of HW time stamp, e.g. based on current drift
	timeStampPtr	current time stamp
Return value:	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description:	Returns a time value out of the HW registers according to the capability of the HW. Is the HW resolution is lower than the Eth_TimeStampType resolution resp. range, than an the remaining bits will be filled with 0. Important Note: Eth_GetCurrentTime may be called within an exclusive area.	
Available via:	`Eth.h`	

] ()

[SWS_Eth_00182] [
If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT.]()

[SWS_Eth_00183] ⌈
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX. ⌋()

[SWS_Eth_00184] ⌈
If development error detection is enabled: the function shall check the parameter timeQualPtr and timeStampPtr for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER. ⌋()

[SWS_Eth_00210] ⌈
The function shall be pre compile time configurable On/Off by the configuration parameter: EthGlobalTimeSupport. ⌋()

[SWS_Eth_00185] ⌈
Caveat: The function requires previous controller initialization (Eth_Init). ⌋()

8.3.14 Eth_EnableEgressTimeStamp

[SWS_Eth_00186] ⌈

Service name:	Eth_EnableEgressTimeStamp	

Syntax:	`void Eth_EnableEgressTimeStamp(`
	`  uint8 CtrlIdx,`
	`  Eth_BufIdxType BufIdx`
	`)`	

Service ID[hex]:	0x17	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the addresses ETH controller.
	BufIdx	Index of the message buffer, where Application expects egress time stamping
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Activates egress time stamping on a dedicated message object. Some HW does store once the egress time stamp marker and some HW needs it always before transmission. There will be no "disable" functionality, due to the fact, that the message type is always "time stamped" by network design.	
Available via:	`Eth.h`	

⌋ ()

[SWS_Eth_00187] ⌈
If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT. ⌋()

[SWS_Eth_00188] ⌈

If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.]()

[SWS_Eth_00211] [
The function shall be pre compile time configurable On/Off by the configuration parameter: EthGlobalTimeSupport.]()

[SWS_Eth_00189] [
Caveat: The function requires previous controller initialization (Eth_Init).]()

8.3.15 Eth_GetEgressTimeStamp

[SWS_Eth_00190] [

Service name:	Eth_GetEgressTimeStamp	

Syntax:	`Std_ReturnType Eth_GetEgressTimeStamp(`
	`  uint8 CtrlIdx,`
	`  Eth_BufIdxType BufIdx,`
	`  Eth_TimeStampQualType* timeQualPtr,`
	`  Eth_TimeStampType* timeStampPtr`
	`)`	

Service ID[hex]:	0x18	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the addresses ETH controller.
	BufIdx	Index of the message buffer, where Application expects egress time stamping
Parameters (inout):	None	
Parameters (out):	timeQualPtr	quality of HW time stamp, e.g. based on current drift
	timeStampPtr	current time stamp
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: failed to read time stamp.
Description:	Reads back the egress time stamp on a dedicated message object. It must be called within the TxConfirmation() function.	
Available via:	`Eth.h`	

] ()

[SWS_Eth_00191] [
If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT.]()

[SWS_Eth_00192] [
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.]()

[SWS_Eth_00193] [
If development error detection is enabled: the function shall check the parameter timeQualPtr and timeStampPtr for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER.]()

[SWS_Eth_00212] ⌈
The function shall be pre compile time configurable On/Off by the configuration parameter: EthGlobalTimeSupport. ⌋()

[SWS_Eth_00194] ⌈
Caveat: The function requires previous controller initialization (Eth_Init). ⌋()

8.3.16 Eth_GetIngressTimeStamp

[SWS_Eth_00195] ⌈

Service name:	Eth_GetIngressTimeStamp	
Syntax:	`Std_ReturnType Eth_GetIngressTimeStamp( uint8 CtrlIdx, const Eth_DataType* DataPtr, Eth_TimeStampQualType* timeQualPtr, Eth_TimeStampType* timeStampPtr )`	
Service ID[hex]:	0x19	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the addresses ETH controller.
	DataPtr	Pointer to the message buffer, where Application expects ingress time stamping
Parameters (inout):	None	
Parameters (out):	timeQualPtr	quality of HW time stamp, e.g. based on current drift
	timeStampPtr	current time stamp
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: failed to read time stamp.
Description:	Reads back the ingress time stamp on a dedicated message object. It must be called within the RxIndication() function.	
Available via:	`Eth.h`	

⌋ ()

[SWS_Eth_00196] ⌈
If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT. ⌋()

[SWS_Eth_00197] ⌈
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX. ⌋()

[SWS_Eth_00198] ⌈
If development error detection is enabled: the function shall check the parameter DataPtr, timeQualPtr and timeStampPtr for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER. ⌋()

[SWS_Eth_00213] ⌈
The function shall be pre compile time configurable On/Off by the configuration parameter: EthGlobalTimeSupport. ⌋()

[SWS_Eth_00199] ⌈
Caveat: The function requires previous controller initialization (Eth_Init). ⌋()

8.3.17 Eth_ProvideTxBuffer

[SWS_Eth_00077] ⌈

Service name:	Eth_ProvideTxBuffer	
Syntax:	`BufReq_ReturnType Eth_ProvideTxBuffer( uint8 CtrlIdx, uint8 Priority, Eth_BufIdxType* BufIdxPtr, uint8** BufPtr, uint16* LenBytePtr )`	
Service ID[hex]:	0x09	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	CtrlIdx	Index of the controller within the context of the Ethernet Driver
	Priority	Frame priority for transmit buffer FIFO selection
Parameters (inout):	LenBytePtr	In: desired length in bytes, out: granted length in bytes
Parameters (out):	BufIdxPtr	Index to the granted buffer resource. To be used for subsequent requests
	BufPtr	Pointer to the granted buffer
Return value:	BufReq_ReturnType	BUFREQ_OK: success BUFREQ_E_NOT_OK: development error detected BUFREQ_E_BUSY: all buffers in use BUFREQ_E_OVFL: requested buffer too large
Description:	Provides access to a transmit buffer of the FIFO related to the specified priority	
Available via:	`Eth.h`	

⌋ () [SWS_Eth_00078] ⌈
The function shall provide a transmit buffer resource. The Ethernet Driver shall lock the buffer until it receives a subsequent call of Eth_Transmit service with the buffer index returned in the BufIdxPtr parameter. ⌋()

[SWS_Eth_00137] ⌈
All locked transmit buffers shall be released if the controller is disabled via Eth_SetControllerMode. ⌋()

[SWS_Eth_00079] ⌈
If a buffer requested with Eth_ProvideTxBuffer that is larger than the available buffer length, the buffer shall not be locked but return the available length and BUFREQ_E_OVFL. ⌋()

[SWS_Eth_00080] ⌈
If all available buffers are in use the component shall return BUFREQ_E_BUSY. ⌋()

[SWS_Eth_00081] ⌈
If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT and return BUFREQ_E_NOT_OK. ⌋()

[SWS_Eth_00082] ⌈
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX and return BUFREQ_E_NOT_OK. ⌋()

[SWS_Eth_00083] ⌈
If development error detection is enabled: the function shall check the parameter BufIdxPtr for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER and return BUFREQ_E_NOT_OK. ⌋()

[SWS_Eth_00084] ⌈
If development error detection is enabled: the function shall check the parameter BufPtr for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER and return BUFREQ_E_NOT_OK. ⌋()

[SWS_Eth_00085] ⌈
If development error detection is enabled: the function shall check the parameter LenBytePtr for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER and return BUFREQ_E_NOT_OK. ⌋()

[SWS_Eth_00086] ⌈
Caveat: The function requires previous controller initialization (Eth_Init). ⌋()

8.3.18 Eth_Transmit

[SWS_Eth_00087] ⌈

Service name:	Eth_Transmit	

Syntax:	`Std_ReturnType Eth_Transmit(`
	`  uint8 CtrlIdx,`
	`  Eth_BufIdxType BufIdx,`
	`  Eth_FrameType FrameType,`
	`  boolean TxConfirmation,`
	`  uint16 LenByte,`
	`  const uint8* PhysAddrPtr`
	`)`	

Service ID[hex]:	0xA	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different buffer indexes and Ctrl indexes	
Parameters (in):	CtrlIdx	Index of the controller within the context of the Ethernet Driver
	BufIdx	Index of the buffer resource
	FrameType	Ethernet frame type
	TxConfirmation	Activates transmission confirmation
	LenByte	Data length in byte
	PhysAddrPtr	Physical target address (MAC address) in network byte order
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: transmission failed
Description:	Triggers transmission of a previously filled transmit buffer	
Available via:	`Eth.h`	

⌋ () [SWS_Eth_00088] ⌈
The function shall build the Ethernet header with the given physical target address (MAC address) and trigger the transmission of a previously filled transmit buffer. ⌋()

After transmission, the driver needs to release the allocated buffer. It is up to the implementation when the actual buffer release shall occur, e.g. within the context of the Eth_TxConfirmation, the Eth_MainFunction, or during the next Eth_ProvideTxBuffer.

[SWS_Eth_00138] ⌈
All pending transmit buffers shall be released if the controller is disabled via Eth_SetControllerMode. ⌋()

[SWS_Eth_00090] ⌈
If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT otherwise (if DET is disabled) return E_NOT_OK. ⌋()

[SWS_Eth_00091] ⌈
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX otherwise (if DET is disabled) return E_NOT_OK. ⌋()

[SWS_Eth_00092] ⌈
If development error detection is enabled: the function shall check the parameter BufIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_PARAM otherwise (if DET is disabled) return E_NOT_OK. ⌋()

[SWS_Eth_00093] ⌈
If development error detection is enabled: the function shall check the parameter PhysAddrPtr for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER otherwise (if DET is disabled) return E_NOT_OK. ⌋()

[SWS_Eth_00129] ⌈
If development error detection is enabled: the function shall check the controller mode for being active (ETH_MODE_ACTIVE). If the check fails, the function shall raise the development error ETH_E_INV_MODE otherwise (if DET is disabled) return E_NOT_OK. ⌋()

[SWS_Eth_00094] ⌈
Caveat: The function requires previous buffer request (Eth_ProvideTxBuffer). ⌋()

8.3.19 Eth_Receive

[SWS_Eth_00095] ⌈

Service name:	Eth_Receive
Syntax:	`void Eth_Receive(` `  uint8 CtrlIdx,`

	uint8 FifoIdx, Eth_RxStatusType* RxStatusPtr)	
Service ID[hex]:	0xB	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different FIFOs. Non Reentrant for the same FIFO.	
Parameters (in):	CtrlIdx	Index of the controller within the context of the Ethernet Driver
	FifoIdx	Specifies the related fifo
Parameters (inout):	None	
Parameters (out):	RxStatusPtr	Indicates whether a frame has been received and if so, whether more frames are available for the related fifo.
Return value:	None	
Description:	Receive a frame from the related fifo.	
Available via:	Eth.h	

] () [SWS_Eth_00096] [
The function shall read the next frame from the receive buffers. The function passes the received frame to the Ethernet interface using the callback function EthIf_RxIndication and indicates if there are more frames in the receive buffers.]()

[SWS_Eth_00097] [
If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT.]()

[SWS_Eth_00098] [
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.]()

[SWS_Eth_00132] [
If development error detection is enabled: the function shall check the controller mode for being active (ETH_MODE_ACTIVE). If the check fails, the function shall raise the development error ETH_E_INV_MODE.]()

[SWS_Eth_00153] [
When calling the callback function EthIf_RxIndication broadcast frames shall be indicated to the Ethernet Interface (see [6]).]()

[SWS_Eth_00099] [
Caveat: The function requires previous controller initialization (Eth_Init).]()

8.3.20 Eth_TxConfirmation

[SWS_Eth_00100] [

Service name:	Eth_TxConfirmation
Syntax:	void Eth_TxConfirmation(uint8 CtrlIdx)
Service ID[hex]:	0xC
Sync/Async:	Synchronous

Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the controller within the context of the Ethernet Driver
Parameters (inout):	None	
Parameters (out):	None	
Return value:	void	None
Description:	Triggers frame transmission confirmation	
Available via:	`Eth.h`	

] () [SWS_Eth_00101] ⌈
The function shall check all filled transmit buffers for successful transmission. The function issues transmit confirmation for each transmitted frame using the callback function EthIf_TxConfirmation if requested by the previous call of Eth_Transmit service. ⌋()

[SWS_Eth_00102] ⌈
If transmission confirmation was enabled by a previous call to Eth_Transmit function the function shall release the buffer resource. ⌋()

[SWS_Eth_00103] ⌈
If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT. ⌋()

[SWS_Eth_00104] ⌈
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX. ⌋()

[SWS_Eth_00134] ⌈
If development error detection is enabled: the function shall check the controller mode for being active (ETH_MODE_ACTIVE). If the check fails, the function shall raise the development error ETH_E_INV_MODE. ⌋()

[SWS_Eth_00105] ⌈
Caveat: The function requires previous initialization (Eth_Init). ⌋()

8.3.21 Eth_GetVersionInfo

[SWS_Eth_00106] ⌈

Service name:	Eth_GetVersionInfo	

Syntax:	`void Eth_GetVersionInfo(`
	`  Std_VersionInfoType* VersionInfoPtr`
	`)`	

Service ID[hex]:	0xD	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	VersionInfoPtr	Version information of this module
Return value:	void	None

Description:	Returns the version information of this module
Available via:	`Eth.h`

] ()

[SWS_Eth_00136] ⌈
If development error detection is enabled: the function shall check the parameter VersionInfoPtr for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER. ⌋()

8.4 Callback notifications

The Ethernet Driver does not provide any callback functions.

8.5 Scheduled functions

8.5.1 Eth_MainFunction

[SWS_Eth_00171] ⌈

Service name:	Eth_MainFunction

Syntax:	`void Eth_MainFunction(`
	`  void`
	`)`

Service ID[hex]:	0x20
Description:	The function checks for controller errors and lost frames. Used for polling state changes. Calls EthIf_CtrlModeIndication when the controller mode changed.
Available via:	`SchM_Eth.h`

] ()

[SWS_Eth_00169] ⌈
The function shall check for lost frames. If the check fails, the function shall raise the extended production error event ETH_E_RX_FRAMES_LOST. ⌋()

[SWS_Eth_00172] ⌈
The function shall check for controller errors (e.g. CRC errors). If the check fails, the function shall raise the extended production error event as defined in section 7.2.2 Extended Production Errors (e.g. ETH_E_CRC). ⌋()

[SWS_Eth_00240] ⌈
Used for polling state changes. Calls EthIf_CtrlModeIndication when the controller mode changed. ⌋()

8.6 Expected Interfaces

This chapter lists all interfaces required from other modules.

8.6.1 Mandatory Interfaces

This chapter defines all interfaces required to fulfill the core functionality of the module.

[SWS_Eth_00119] ⌈

API function	*Header File*	*Description*
Dem_SetEventStatus	Dem.h	Called by SW-Cs or BSW modules to report monitor status information to the Dem. BSW modules calling Dem_SetEventStatus can safely ignore the return value.
EthIf_CtrlModeIndication	EthIf.h	Called asynchronously when mode has been read out. Triggered by previous Eth_SetControllerMode call. Can directly be called within the trigger functions.
EthIf_RxIndication	EthIf.h	Handles a received frame received by the indexed controller
EthIf_TxConfirmation	EthIf.h	Confirms frame transmission by the indexed controller
SchM_Enter_Eth	SchM_<Mip>.h	Invokes the SchM_Enter function to enter a module local exclusive area.
SchM_Exit_Eth	SchM_<Mip>.h	Invokes the SchM_Exit function to exit an exclusive area.

⌋ ()

8.6.2 Optional Interfaces

This chapter defines all interfaces required to fulfill an optional functionality of the module.

[SWS_Eth_00120] ⌈

API function	*Header File*	*Description*
Det_ReportError	Det.h	Service to report development errors.
EthSwt_EthRxFinishedIndication	EthSwt_Eth.h	Indication for a finished receive process for a specific Ethernet frame, which results in providing the management information retrieved during EthSwt_EthRxProcessFrame().
EthSwt_EthRxProcessFrame	EthSwt_Eth.h	Function inspects the Ethernet frame passed by the data pointer for management information and stores it for later use in EthSwt_EthRxFinishedIndication().
EthSwt_EthTxAdaptBufferLength	EthSwt_Eth.h	Modifies the buffer length to be able to insert management information.
EthSwt_EthTxFinishedIndication	EthSwt_Eth.h	Indication for a finished transmit process for a specific Ethernet frame.
EthSwt_EthTxPrepareFrame	EthSwt_Eth.h	Prepares the Ethernet frame for common Ethernet communication (frame shall be handled by switch according to the common address resolution behavior) and stores the information for processing of EthSwt_EthTxFinishedIndication().
EthSwt_EthTxProcessFrame	EthSwt_Eth.h	Function inserts management information into the Ethernet frame.

⌋ ()

8.6.3 Configurable interfaces

The Ethernet Driver does not use configurable interfaces.

Terms and definitions:
Reentrant: interface is expected to be reentrant
Don't care: reentrancy of interface not relevant for this module (in general it is in this case not reentrant).

9 Sequence diagrams

The usage of the Ethernet Driver is depicted in the sequence diagrams of the Ethernet Interface.

10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification Chapter 10.1 describes fundamentals. It also specifies a template (table) you shall use for the parameter specification. We intend to leave Chapter 10.1 in the specification to guarantee comprehension.

Chapter 10.2 specifies the structure (containers) and the parameters of the module Ethernet Driver.

Chapter 10.3 specifies published information of the module Ethernet Driver.
.

10.1 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapters 7 and Chapter 8.

[SWS_Eth_00257] DRAFT ⌈
The Ethernet Driver module shall reject configurations with partition mappings which are not supported by the implementation.⌋ ()

[SWS_Eth_00258] DRAFT ⌈
If the driver manages several Ethernet controllers and if a subset of these controllers share peripheral resources or are somehow coupled (E.g. Communication control can only be done globally for all controllers), Ethernet driver shall emulate independent controllers to the upper layers. The coordination (E.g. Communication control) has to be done by the upper layer modules.⌋ ()

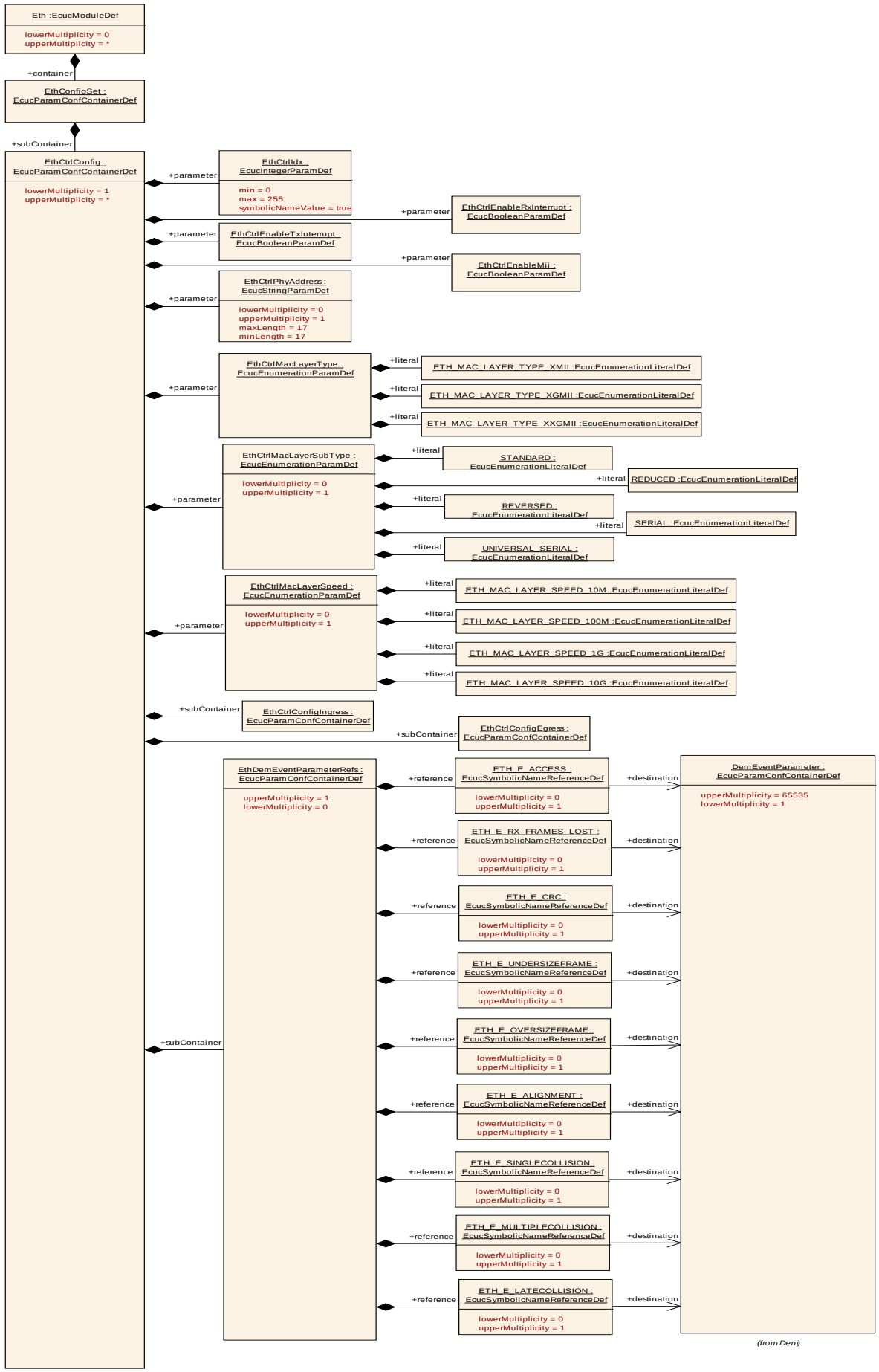
Eth :EcucModuleDef
lowerMultiplicity = 0
upperMultiplicity = *
+container
EthConfigSet :
EcucParamConfContainerDef
+subContainer
EthCtrlConfig :
EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+parameter
EthCtrlIdx :
EcucIntegerParamDef
min = 0
max = 255
symbolicNameValue = true
EthCtrlEnableRxInterrupt :
EcucBooleanParamDef
EthCtrlEnableTxInterrupt :
EcucBooleanParamDef
EthCtrlEnableMii :
EcucBooleanParamDef
EthCtrlPhyAddress :
EcucStringParamDef
lowerMultiplicity = 0
upperMultiplicity = 1
maxLength = 17
minLength = 17
EthCtrlMacLayerType :
EcucEnumerationParamDef
+literal
ETH_MAC_LAYER_TYPE_XMII :EcucEnumerationLiteralDef
ETH_MAC_LAYER_TYPE_XGMII :EcucEnumerationLiteralDef
ETH_MAC_LAYER_TYPE_XXGMII :EcucEnumerationLiteralDef
EthCtrlMacLayerSubType :
EcucEnumerationParamDef
lowerMultiplicity = 0
upperMultiplicity = 1
STANDARD :
EcucEnumerationLiteralDef
REDUCED :EcucEnumerationLiteralDef
REVERSED :
EcucEnumerationLiteralDef
SERIAL :EcucEnumerationLiteralDef
UNIVERSAL_SERIAL :
EcucEnumerationLiteralDef
EthCtrlMacLayerSpeed :
EcucEnumerationParamDef
lowerMultiplicity = 0
upperMultiplicity = 1
ETH_MAC_LAYER_SPEED_10M :EcucEnumerationLiteralDef
ETH_MAC_LAYER_SPEED_100M :EcucEnumerationLiteralDef
ETH_MAC_LAYER_SPEED_1G :EcucEnumerationLiteralDef
ETH_MAC_LAYER_SPEED_10G :EcucEnumerationLiteralDef
+subContainer
EthCtrlConfigIngress :
EcucParamConfContainerDef
EthCtrlConfigEgress :
EcucParamConfContainerDef
EthDemEventParameterRefs :
EcucParamConfContainerDef
upperMultiplicity = 1
lowerMultiplicity = 0
+reference
ETH_E_ACCESS :
EcucSymbolicNameReferenceDef
lowerMultiplicity = 0
upperMultiplicity = 1
ETH_E_RX_FRAMES_LOST :
EcucSymbolicNameReferenceDef
ETH_E_CRC :
EcucSymbolicNameReferenceDef
ETH_E_UNDERSIZEFRAME :
EcucSymbolicNameReferenceDef
ETH_E_OVERSIZEFRAME :
EcucSymbolicNameReferenceDef
ETH_E_ALIGNMENT :
EcucSymbolicNameReferenceDef
ETH_E_SINGLECOLLISION :
EcucSymbolicNameReferenceDef
ETH_E_MULTIPLECOLLISION :
EcucSymbolicNameReferenceDef
ETH_E_LATECOLLISION :
EcucSymbolicNameReferenceDef
+destination
DemEventParameter :
EcucParamConfContainerDef
upperMultiplicity = 65535
lowerMultiplicity = 1
(from Dem)

Figure 10.1: Ethernet Driver configuration structure

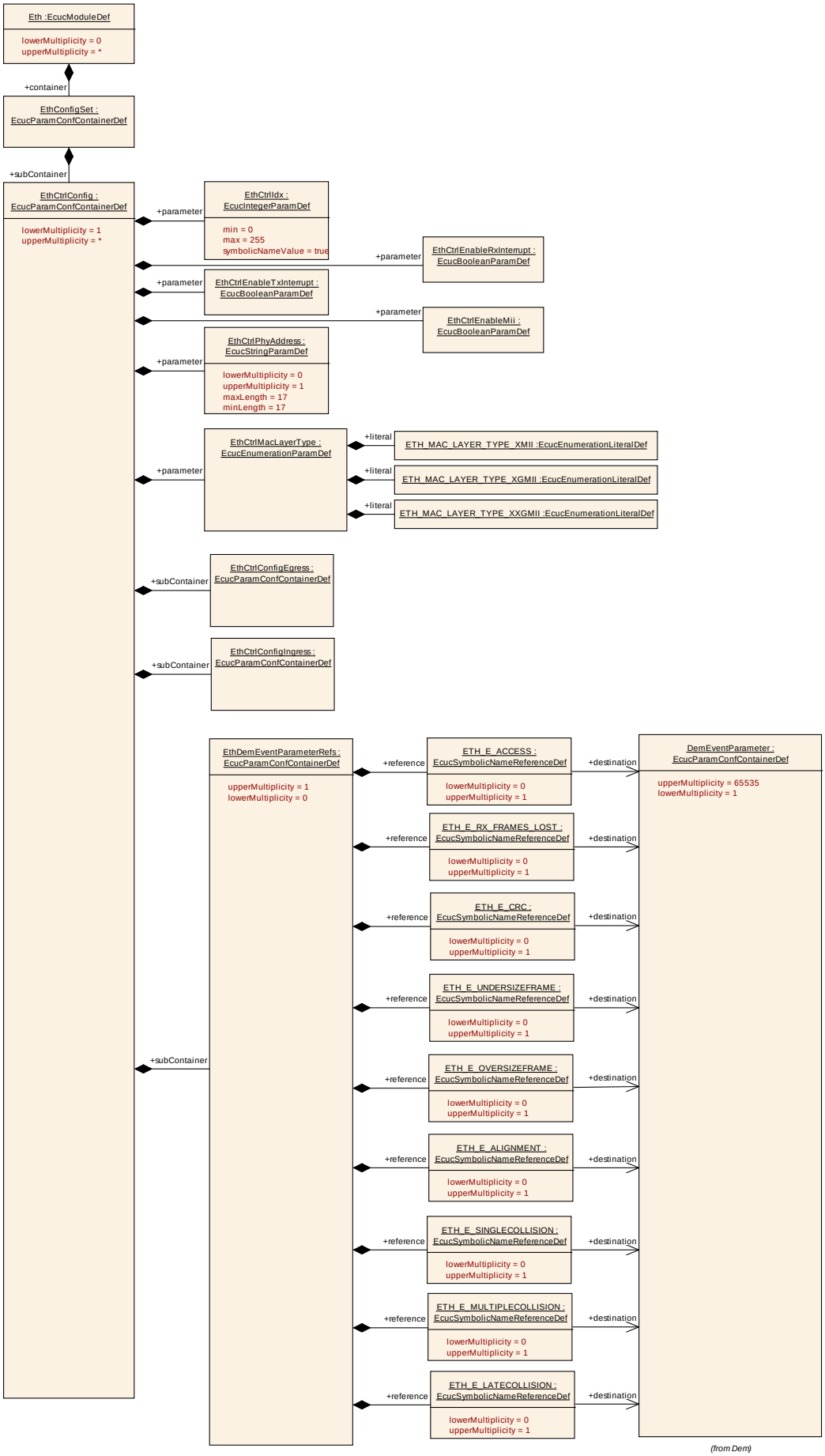
Eth :EcucModuleDef
lowerMultiplicity = 0
upperMultiplicity = *
+container
EthConfigSet :
EcucParamConfContainerDef
+subContainer
EthCtrlConfig :
EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+parameter
EthCtrlIdx :
EcucIntegerParamDef
min = 0
max = 255
symbolicNameValue = true
+parameter
EthCtrlEnableRxInterrupt :
EcucBooleanParamDef
+parameter
EthCtrlEnableTxInterrupt :
EcucBooleanParamDef
+parameter
EthCtrlEnableMii :
EcucBooleanParamDef
+parameter
EthCtrlPhyAddress :
EcucStringParamDef
lowerMultiplicity = 0
upperMultiplicity = 1
maxLength = 17
minLength = 17
+parameter
EthCtrlMacLayerType :
EcucEnumerationParamDef
+literal
ETH_MAC_LAYER_TYPE_XMII :EcucEnumerationLiteralDef
+literal
ETH_MAC_LAYER_TYPE_XGMII :EcucEnumerationLiteralDef
+literal
ETH_MAC_LAYER_TYPE_XXGMII :EcucEnumerationLiteralDef
+subContainer
EthCtrlConfigEgress :
EcucParamConfContainerDef
+subContainer
EthCtrlConfigIngress :
EcucParamConfContainerDef
+subContainer
EthDemEventParameterRefs :
EcucParamConfContainerDef
upperMultiplicity = 1
lowerMultiplicity = 0
+reference
ETH_E_ACCESS :
EcucSymbolicNameReferenceDef
lowerMultiplicity = 0
upperMultiplicity = 1
+destination
DemEventParameter :
EcucParamConfContainerDef
upperMultiplicity = 65535
lowerMultiplicity = 1
+reference
ETH_E_RX_FRAMES_LOST :
EcucSymbolicNameReferenceDef
lowerMultiplicity = 0
upperMultiplicity = 1
+destination
+reference
ETH_E_CRC :
EcucSymbolicNameReferenceDef
lowerMultiplicity = 0
upperMultiplicity = 1
+destination
+reference
ETH_E_UNDERSIZEFRAME :
EcucSymbolicNameReferenceDef
lowerMultiplicity = 0
upperMultiplicity = 1
+destination
+reference
ETH_E_OVERSIZEFRAME :
EcucSymbolicNameReferenceDef
lowerMultiplicity = 0
upperMultiplicity = 1
+destination
+reference
ETH_E_ALIGNMENT :
EcucSymbolicNameReferenceDef
lowerMultiplicity = 0
upperMultiplicity = 1
+destination
+reference
ETH_E_SINGLECOLLISION :
EcucSymbolicNameReferenceDef
lowerMultiplicity = 0
upperMultiplicity = 1
+destination
+reference
ETH_E_MULTIPLECOLLISION :
EcucSymbolicNameReferenceDef
lowerMultiplicity = 0
upperMultiplicity = 1
+destination
+reference
ETH_E_LATECOLLISION :
EcucSymbolicNameReferenceDef
lowerMultiplicity = 0
upperMultiplicity = 1
+destination
(from Dem)

Figure 10.2: Ethernet Driver Controller configuration structure

10.1.1 Eth

SWS Item	**ECUC_Eth_00038 :**
Module Name	*Eth*
Module Description	Configuration of the Eth (Ethernet Driver) module.
Post-Build Variant Support	true
Supported Config Variants	VARIANT-LINK-TIME, VARIANT-POST-BUILD, VARIANT-PRE-COMPILE

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
EthConfigSet	1	This container contains the configuration parameters and sub containers of the AUTOSAR Eth module.
EthGeneral	1	General configuration of Ethernet Driver module

10.1.2 EthConfigSet

SWS Item	**ECUC_Eth_00015 :**
Container Name	EthConfigSet
Description	This container contains the configuration parameters and sub containers of the AUTOSAR Eth module.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
EthCtrlConfig	1..*	Configuration of the individual controller

10.1.3 EthCtrlConfig

SWS Item	**ECUC_Eth_00006 :**
Container Name	EthCtrlConfig
Description	Configuration of the individual controller
Configuration Parameters	

SWS Item	**ECUC_Eth_00012 :**		
Name	EthCtrlEnableMii		
Parent Container	EthCtrlConfig		
Description	Enables / Disables Media Independent Interface (MII) for transceiver access		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	ECUC_Eth_00010 :		
Name	EthCtrlEnableRxInterrupt		
Parent Container	EthCtrlConfig		
Description	Enables / Disables receive interrupt		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	ECUC_Eth_00011 :		
Name	EthCtrlEnableTxInterrupt		
Parent Container	EthCtrlConfig		
Description	Enables / Disables transmit interrupt		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	ECUC_Eth_00007 :		
Name	EthCtrlIdx		
Parent Container	EthCtrlConfig		
Description	Specifies the instance ID of the configured controller.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	ECUC_Eth_00063 :		
Name	EthCtrlMacLayerSpeed		
Parent Container	EthCtrlConfig		
Description	Defines the baud rate of the MAC layer.		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	ETH_MAC_LAYER_SPEED_100M	--	
	ETH_MAC_LAYER_SPEED_10G	--	
	ETH_MAC_LAYER_SPEED_10M	--	
	ETH_MAC_LAYER_SPEED_1G	--	
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity	Pre-compile time	X	VARIANT-PRE-COMPILE

Configuration Class	***Link time***	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Eth_00062 :**		
Name	EthCtrlMacLayerSubType		
Parent Container	EthCtrlConfig		
Description	Defines the MAC layer subtype of a switch port		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	REDUCED	--	
	REVERSED	--	
	SERIAL	--	
	STANDARD	--	
	UNIVERSAL_SERIAL	--	
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Eth_00039 :**	
Name	EthCtrlMacLayerType	
Parent Container	EthCtrlConfig	
Description	Defines the MAC layer type of the ethernet controller.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	ETH_MAC_LAYER_TYPE_XGMII	MAC layer interface (data) bandwith class 1Gbit/s (e.g. GMII, RGMII, SGMII, RvGMII, USGMII)
	ETH_MAC_LAYER_TYPE_XMII	MAC layer interface (data) bandwith class 100Mbit/s (e.g. RMII, RvMII, SMII, RvMII)
	ETH_MAC_LAYER_TYPE_XXGMII	MAC layer interface (data) bandwith class 10Gbit/s
Post-Build Variant Value	true	
Value Configuration Class	***Pre-compile time***	X VARIANT-PRE-COMPILE
	Link time	X VARIANT-LINK-TIME
	Post-build time	X VARIANT-POST-BUILD
Scope / Dependency	scope: ECU	

SWS Item	**ECUC_Eth_00020 :**		
Name	EthCtrlPhyAddress		
Parent Container	EthCtrlConfig		
Description	Specifies the unique 48-bit physical address (MAC address) of the controller in network byte order. Regular Expression: [0-9a-fA-F]{2}[[:-][0-9a-fA-F]{2}]{5}		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	17		
minLength	17		
regularExpression	--		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_Eth_00065 :**		
Name	EthCtrlEcucPartitionRef		
Parent Container	EthCtrlConfig		
Description	Maps the Ethernet controller to zero or one ECUC partitions. The ECUC partition referenced is a subset of the ECUC partitions where the Ethernet driver is mapped to. **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	Reference to [EcucPartition]		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
EthCtrlConfigEgress	1	Configuration of one Ethernet controler egress behavior.
EthCtrlConfigIngress	1	Configuration of one Ethernet controler ingress behavior.
EthDemEventParameterRefs	0..1	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_SetEventStatus in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId symbolic value. The standardized errors are provided in this container and can be extended by vendor-specific error references.

[SWS_Eth_00260] DRAFT [
The ECUC partitions referenced by EthCtrlEcucPartitionRef shall be a subset of the ECUC partitions referenced by EthEcucPartitionRef.] ()

[SWS_Eth_00261] DRAFT [
EthCtrlConfig, EthTrcvConfig and EthSwtConfig (if existent in configuration) of one communication channel shall all reference the same ECUC partition] ().

10.1.4 EthCtrlConfigEgress

SWS Item	**ECUC_Eth_00046 :**		
Container Name	EthCtrlConfigEgress		
Description	Configuration of one Ethernet controler egress behavior.		
Configuration Parameters			

SWS Item	**ECUC_Eth_00052 :**		
Name	EthCtrlConfigEgressLastSchedulerRef		
Parent Container	EthCtrlConfigEgress		
Description	Reference to the scheduler which is the last in the egress structure.		
Multiplicity	1		
Type	Reference to [EthCtrlConfigScheduler]		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
EthCtrlConfigEgressFifo	0..*	Represents a Fifo at the egress side.
EthCtrlConfigScheduler	1..*	Represents a Scheduler on the egress side.
EthCtrlConfigShaper	0..*	Represents a Shaper an the egress side.

10.1.5 EthCtrlConfigEgressFifo

SWS Item	**ECUC_Eth_00047 :**		
Container Name	EthCtrlConfigEgressFifo		
Description	Represents a Fifo at the egress side.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

SWS Item	**ECUC_Eth_00051 :**
Name	EthCtrlConfigEgressFifoBufLenByte
Parent Container	EthCtrlConfigEgressFifo
Description	Length of Fifo elements in bytes.
Multiplicity	1
Type	EcucIntegerParamDef

Range	0 .. 65535		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_Eth_00050 :**		
Name	EthCtrlConfigEgressFifoBufTotal		
Parent Container	EthCtrlConfigEgressFifo		
Description	Fifo buffer count.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_Eth_00048 :**		
Name	EthCtrlConfigEgressFifoIdx		
Parent Container	EthCtrlConfigEgressFifo		
Description	Egress Fifo index.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_Eth_00049 :**		
Name	EthCtrlConfigEgressFifoPriorityAssignment		
Parent Container	EthCtrlConfigEgressFifo		
Description	Message egress prority assignment.		
Multiplicity	0..*		
Type	EcucIntegerParamDef		
Range	0 .. 7		
Default value	--		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

10.1.6 EthCtrlConfigScheduler

SWS Item	**ECUC_Eth_00053 :**
Container Name	EthCtrlConfigScheduler
Description	Represents a Scheduler on the egress side.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
EthCtrlConfigSchedulerPredecessor	1..*	Defines an ordered list of predecessors for this scheduler.

10.1.7 EthCtrlConfigSchedulerPredecessor

SWS Item	**ECUC_Eth_00054 :**
Container Name	EthCtrlConfigSchedulerPredecessor
Description	Defines an ordered list of predecessors for this scheduler.
Configuration Parameters	

SWS Item	**ECUC_Eth_00055 :**		
Name	EthCtrlConfigSchedulerPredecessorOrder		
Parent Container	EthCtrlConfigSchedulerPredecessor		
Description	Defines the order of the scheduler predecessors.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 18446744073709551615		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_Eth_00056 :**		
Name	EthCtrlConfigSchedulerPredecessorRef		
Parent Container	EthCtrlConfigSchedulerPredecessor		
Description	Choice reference to the scheduler predecessor.		
Multiplicity	1		
Type	Choice reference to [EthCtrlConfigEgressFifo , EthCtrlConfigScheduler , EthCtrlConfigShaper]		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.1.8 EthCtrlConfigShaper

SWS Item	**ECUC_Eth_00057 :**
Container Name	EthCtrlConfigShaper
Description	Represents a Shaper an the egress side.
Configuration Parameters	

SWS Item	**ECUC_Eth_00058 :**		
Name	EthCtrlConfigShaperIdleSlope		
Parent Container	EthCtrlConfigShaper		
Description	Defines the increase of credit in bits per second for the AVB shaper.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 18446744073709551615		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_Eth_00059 :**		
Name	EthCtrlConfigShaperPredecessorFifoRef		
Parent Container	EthCtrlConfigShaper		
Description	Reference to the fifo which is the predecessor for this shaper.		
Multiplicity	1		
Type	Reference to [EthCtrlConfigEgressFifo]		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.1.9 EthCtrlConfigIngress

SWS Item	**ECUC_Eth_00040 :**
Container Name	EthCtrlConfigIngress
Description	Configuration of one Ethernet controler ingress behavior.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
EthCtrlConfigIngressFifo	0..*	Represents a Fifo at the ingress side.

10.1.10 EthCtrlConfigIngressFifo

SWS Item	**ECUC_Eth_00041 :**
Container Name	EthCtrlConfigIngressFifo

Description	Represents a Fifo at the ingress side.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

SWS Item	**ECUC_Eth_00045 :**		
Name	EthCtrlConfigIngressFifoBufLenByte		
Parent Container	EthCtrlConfigIngressFifo		
Description	Length of Fifo elements in bytes.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_Eth_00044 :**		
Name	EthCtrlConfigIngressFifoBufTotal		
Parent Container	EthCtrlConfigIngressFifo		
Description	Fifo buffer count.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_Eth_00043 :**		
Name	EthCtrlConfigIngressFifoIdx		
Parent Container	EthCtrlConfigIngressFifo		
Description	Ingress Fifo index.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_Eth_00042 :**		
Name	EthCtrlConfigIngressFifoPriorityAssignment		
Parent Container	EthCtrlConfigIngressFifo		
Description	Message ingress prority assignment.		
Multiplicity	0..*		

Type	EcucIntegerParamDef		
Range	0 .. 7		
Default value	--		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

10.1.11 EthDemEventParameterRefs

SWS Item	**ECUC_Eth_00016 :**
Container Name	EthDemEventParameterRefs
Description	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_SetEventStatus in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId symbolic value. The standardized errors are provided in this container and can be extended by vendor-specific error references.
Configuration Parameters	

SWS Item	**ECUC_Eth_00017 :**		
Name	ETH_E_ACCESS		
Parent Container	EthDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when the error "Controller access failed" has occured.		
Multiplicity	0..1		
Type	Symbolic name reference to [DemEventParameter]		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_Eth_00026 :**
Name	ETH_E_ALIGNMENT
Parent Container	EthDemEventParameterRefs
Description	Reference to the DemEventParameter which shall be issued when the error "Alignment Error" has occured.
Multiplicity	0..1
Type	Symbolic name reference to [DemEventParameter]

Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_Eth_00023 :**		
Name	ETH_E_CRC		
Parent Container	EthDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when the error "CRC Failure" has occured.		
Multiplicity	0..1		
Type	Symbolic name reference to [DemEventParameter]		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_Eth_00029 :**		
Name	ETH_E_LATECOLLISION		
Parent Container	EthDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when the error "Late Collisions" has occured.		
Multiplicity	0..1		
Type	Symbolic name reference to [DemEventParameter]		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_Eth_00028 :**		
Name	ETH_E_MULTIPLECOLLISION		
Parent Container	EthDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when the error "Multiple Collisions" has occured.		
Multiplicity	0..1		
Type	Symbolic name reference to [DemEventParameter]		
Post-Build Variant Multiplicity	true		

Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_Eth_00025 :**		
Name	ETH_E_OVERSIZEFRAME		
Parent Container	EthDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when the error "Oversized Frame" has occured.		
Multiplicity	0..1		
Type	Symbolic name reference to [DemEventParameter]		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_Eth_00021 :**		
Name	ETH_E_RX_FRAMES_LOST		
Parent Container	EthDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when the error "receive frames lost" has occured.		
Multiplicity	0..1		
Type	Symbolic name reference to [DemEventParameter]		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_Eth_00027 :**		
Name	ETH_E_SINGLECOLLISION		
Parent Container	EthDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when the error "Single Collisions" has occured.		
Multiplicity	0..1		
Type	Symbolic name reference to [DemEventParameter]		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration	***Pre-compile time***	X	VARIANT-PRE-COMPILE

Class	***Link time***	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_Eth_00024 :**		
Name	ETH_E_UNDERSIZEFRAME		
Parent Container	EthDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when the error "Undersized Frame" has occured.		
Multiplicity	0..1		
Type	Symbolic name reference to [DemEventParameter]		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

10.1.12 EthGeneral

SWS Item	**ECUC_Eth_00001 :**
Container Name	EthGeneral
Description	General configuration of Ethernet Driver module
Configuration Parameters	

SWS Item	**ECUC_Eth_00003 :**		
Name	EthDevErrorDetect		
Parent Container	EthGeneral		
Description	Switches the development error detection and notification on or off. • true: detection and notification is enabled. • false: detection and notification is disabled.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Eth_00035 :**
Name	EthGetDropCountApi

Parent Container	EthGeneral		
Description	Enables / Disables Eth_GetCounterValues API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Eth_00036 :**		
Name	EthGetEtherStatsApi		
Parent Container	EthGeneral		
Description	Enables / Disables Eth_GetEtherStats API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Eth_00061 :**		
Name	EthGetTxErrorCounterValuesApi		
Parent Container	EthGeneral		
Description	Enables/Disables Eth_GetTxErrorCounterValues API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Eth_00060 :**		
Name	EthGetTxStatsApi		
Parent Container	EthGeneral		
Description	Enables/Disables Eth_GetTxStats API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Eth_00037 :**
Name	EthGlobalTimeSupport
Parent Container	EthGeneral
Description	Enables/Disables the GlobalTime APIs used amongst others by Global Time Synchronization over Ethernet.

Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Eth_00018 :**		
Name	EthIndex		
Parent Container	EthGeneral		
Description	Specifies the InstanceId of this module instance. If only one instance is present it shall have the Id 0.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Eth_00022 :**		
Name	EthMainFunctionPeriod		
Parent Container	EthGeneral		
Description	Specifies the period of main function Eth_MainFunction in seconds. Ethernet driver does not require this information but the BSW scheduler.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	]0 .. INF[		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Eth_00002 :**		
Name	EthMaxCtrlsSupported		
Parent Container	EthGeneral		
Description	Limits the total number of supported controllers.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 255		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Eth_00004 :**
Name	EthVersionInfoApi
Parent Container	EthGeneral

Description	Enables / Disables version info API		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Eth_00064 :**		
Name	EthEcucPartitionRef		
Parent Container	EthGeneral		
Description	Maps the Ethernet driver to zero or multiple ECUC partitions to make the modules API available in this partition. The Ethernet driver will operate as an independent instance in each of the partitions. **Tags:** atp.Status=draft		
Multiplicity	0..*		
Type	Reference to [EcucPartition]		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
EthCtrlOffloading	1	Configuration of hardware offloading features.

[SWS_Eth_00259] DRAFT [
The module will operate as an independent instance in each of the partitions, means the called API will only target the partition it is called in.]()

10.1.13 EthCtrlOffloading

SWS Item	**ECUC_Eth_00030 :**
Container Name	EthCtrlOffloading
Description	Configuration of hardware offloading features.
Configuration Parameters	

SWS Item	**ECUC_Eth_00032 :**
Name	EthCtrlEnableOffloadChecksumICMP
Parent Container	EthCtrlOffloading
Description	Enables / Disables hardware offloading for ICMP checksums.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	--
Post-Build Variant Value	false

Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Eth_00031 :**		
Name	EthCtrlEnableOffloadChecksumIPv4		
Parent Container	EthCtrlOffloading		
Description	Enables / Disables hardware offloading for IPv4 checksums.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Eth_00033 :**		
Name	EthCtrlEnableOffloadChecksumTCP		
Parent Container	EthCtrlOffloading		
Description	Enables / Disables hardware offloading for TCP checksums.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Eth_00034 :**		
Name	EthCtrlEnableOffloadChecksumUDP		
Parent Container	EthCtrlOffloading		
Description	Enables / Disables hardware offloading for UDP checksums.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

11 Not applicable requirements

[SWS_Eth_00999]
These requirements are not applicable to this specification (BSW00170).